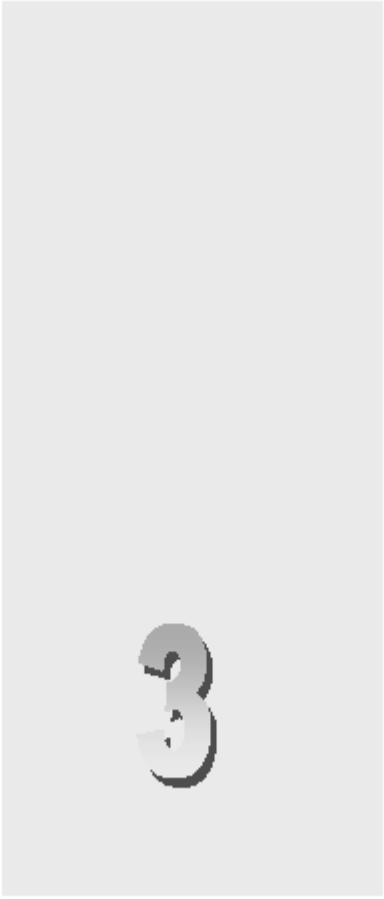

La Programación de *Calcicat*

3.1. Motivos del lenguaje de programación escogido.

Al comienzo del proyecto a realizar se estudiaron diversos lenguajes de programación con la finalidad de escoger aquel cuyas características resultasen óptimas tanto en tiempo de aprendizaje como en tiempo de diseño y desarrollo. Por ello, se pensó en la utilización de lenguajes visuales donde el programador puede crear aplicaciones de consola en tiempos extremadamente breves. Además la posibilidad de generar archivos ejecutables (.exe) que permitiesen la utilización del software en cualquier máquina aun sin disponer del compilador fue fundamental en la elección final. De este modo, y con unos pocos candidatos como Builder C, C/C++, Visual Java, Visual Basic y Visual NET, la balanza se decantó del lado de Visual Basic apelando en gran parte a las razones citadas a continuación.

En primer lugar, Visual Basic (VB) es un lenguaje de programación que despierta gran entusiasmo tanto en programadores expertos como inexpertos. En el caso de los programadores expertos por la facilidad con la que desarrollan aplicaciones complejas en tiempos muy breves en comparación, por ejemplo, con Visual C++. Por otro lado, los programadores novatos se ven motivados por el hecho de ver de lo que son capaces de hacer al poco tiempo de estudio del lenguaje. Si bien todo esto no exime de desventajas a VB ya que el precio a pagar es una menor eficiencia de las aplicaciones y una menor velocidad.

VB es considerado como un lenguaje de programación de cuarta generación, refiriéndose al hecho de que un gran número de aplicaciones o tareas se realizan sin apenas escribir código. Además también es un lenguaje orientado a objetos, aunque no con programación orientada a objetos (POO) como por ejemplo Java. La diferencia de VB con los lenguajes POO es que utiliza objetos con propiedades y

métodos pero careciendo en todo momento de mecanismos de herencia y polimorfismo.

Con la única finalidad de que el lector pueda familiarizarse con el lenguaje de programación citado y que posteriormente sea capaz de comprender los conceptos mencionados en el capítulo, el siguiente punto es un breve manual de referencia del lenguaje VB en su versión 6.0 aunque se han incluido todos los estándares entre las distintas versiones por facilitar el aprendizaje.

3.2. Manual de referencia de Visual Basic 6.0

3.2.1. Elementos del lenguaje

a. Comentarios

Visual Basic interpreta como comentario cualquier frase precedida por el carácter comilla simple (‘) y no ejecuta ninguna acción. Por ejemplo:

‘ Esto es un comentario en Visual Basic.

b. Variables

Las variables pueden ser de cualquiera de los siguientes tipos:

Tipo	Descripción	Carácter de declaración	Rango
Boolean	Binario	/	True o False
Byte	Entero corto	*	0 a 255
Integer	Entero de 2 bytes	%	-32768 a 32767
Long	Entero de 4 bytes	&	-2147483648 a 2147483647
Single	Real de precisión simple (4 bytes)	¡	-3.40e38 a 3.40e38
Double	Real de doble precisión (8 bytes)	#	-1.79D+308 a 1.79D+308
Currency	Número con punto decimal fijo (8bytes)	@	-9.22e+14 a 9.22e+14
String	Cadena de caracteres	$	0 a 65500 caracteres
Date	Fecha (8 bytes)		
Variant	Todos los anteriores	Ninguno	Según su definición
User-Defined	Tipos de datos o estructuras	Ninguno	

c. Declaración de variables

Antes de utilizar cualquier variable es aconsejable inicializar su valor y tipo. Una forma de realizar esto es mediante la sentencia Dim o una de las palabras clave Public, Private o Static. La declaración es como sigue:

Dim Contador *As* Integer : *Dim* Nombre *As* String

'El carácter : permite anidar sentencias en una misma línea de código.

d. Ámbito de las variables

La siguiente tabla muestra el esquema de dónde una variable es accesible por un módulo o submódulo de una aplicación.

Tipo de variable	Lugar de declaración	Accesibilidad
Global o public	*.bas	Desde todos los formularios
Public	*.bas	Desde todas las funciones de ese módulo
Dim o private	*.frm	Desde cualquier procedimiento del propio formulario
Dim	Cualquier procedimiento de un módulo	Desde el propio procedimiento

Para declarar las variables hay que ir a la ventana de código del formulario y seleccionar la pestaña de ***General*** y ***Declaraciones***.

e. Constantes simbólicas

Sirven para recordar valores complicados, para definir cadenas repetitivas, textos... y su sintaxis es:

[Public|Private] Const NombreConstante [As tipo] = Expresión.

f. Operadores

La siguiente tabla muestra un resumen de los principales operadores:

Tipo	Operación	Operador en VBasic
Aritméticos	Exponenciación	^
	Cambio de signo	-
	Multiplicación y división	* , /
	División entera	\

	Resto de división entera	Mod
	Suma y resta	+ , -
Concatenación	Concatenar o enlazar	& +
Relacional	Igual a	=
	Distinto	<>
	Menor que / Menor o igual que	< , <=
	Mayor que/ Mayor o igual que	> , >=
Otros	Comparar caracteres	Like
	Comparar dos referencias a objetos	Is
Lógico	Negación	Not
	And	and
	Or qinclusivo	Or
	Or exclusivo	Xor
	Equivalencia	Eqv
	Implicación	Imp

g. Sentencias

Son líneas de texto que involucran a una o más operaciones a realizar.

h. El tipo Variant

Es un tipo de datos que permite almacenar todos los tipos de datos definidos.

i. El tipo String

Permite almacenar una cadena de caracteres. El ejemplo clarifica el proceso:

Dim Nombre *As* String

Nombre = "Joan Escrivá"

j. El tipo enumerado

Es una lista de valores que pueden ser tomados por una variable de ese tipo.

Public Enum DiasSemana Lunes Martes Miércoles ... End enum	'Declaro variable de tipo DiasSemana. Public hoy As DiasSemana 'Por defecto lunes=0, martes=1,... 'hoy=domingo es equivalente a decir hoy=6

3.2.2 Entrada y salida de datos

Visual Basic provee de funciones para introducir datos como, por ejemplo, ***InputBox*** cuya sintaxis es:

Resp = InputBox(mensaje, [título], [por_omisión],[posición_x],[posición_y]

Como ejemplo de uso, véase la siguiente línea:

Nombre = InputBox("Diga su nombre:"', "Datos", "Joan", 12, 50)

La respuesta se guardará en la variable Nombre.

Interceptando la tecla pulsada.

Cuando el usuario pulsa una tecla ocurren tres eventos: Key-Down, Key-Press y Key-Up. El más general es Key-Press que se genera solamente cuando se introduce un carácter ASCII; no se incluyen las teclas F1 a F12 y las teclas de edición *Ins*, *Supr*, etc. A continuación se muestran funciones relacionadas:

Chr	Convierte a carácter un determinado valor.
UCase(*car*)	Convierte el carácter *car* a mayúsculas.
Asc(*car*)	Devuelve el valor ascii del carácter *car*.

3.2.3 Sentencias de control

Permiten tomar decisiones y realizar una determinada tarea repetidas veces.

IF...THEN...ELSE	SELECT CASE	FOR...NEXT

If condicion then
Sentencia(s)
ElseIf condicion2 then
Sentencia(s)
Else
Sentencia(s)
End If
	Select case expresion
	Case etiqueta1
	Sentencia(s)
	Case etiqueta2
	Sentecnia(s)
	Case else
	Sentencias(s)
	End Select
		For *var=Ini* To *var=Fin*
		Sentencia(s)
		Exit For
		Sentencia(s)
		Next [variable]

DO...LOOP	**WHILE...WEND**	**FOR EACH...NEXT**
Do [{*while/Until*}] Cond Sentencia(s) Exit Do Sentencia(s) Loop [{*while/Until*}]	While condicion Sentencia(s) Wend	For each var In grupo Sentencia(s) Next variable

DO...LOOP presenta dos formatos, según donde figura el [{*while/Until*}]

También existe la sentencia GOTO que transfiere el control a una línea específica del código. Su sintaxis y utilización se muestran a continuación:

Goto {etiqueta/nº_de_línea}	Comienzo: Expresiones... Goto Comienzo

3.2.4 Estructuras de datos

Hay dos estructuras de datos en Visual Basic: registro o estructura y matriz. La matriz es un conjunto de datos todos del mismo tipo que comparten un nombre común mientras que la estructura o registro es un nuevo tipo de datos declarado por el usuario.

3.2.4.1 Matrices (Declaración)

Pueden definirse matrices de *n* dimensiones tal y como se muestra a continuación:

Dim MatUnidim(69 *) As Integer*	Se crea una matriz unidimensional de 70 elementos de tipo entero.
Dim MatBid(10, 11) *As Double*	Declaración de matriz bidimensional de tamaño 10*11 de tipos Double.
Dim Mat()	Creación de una matriz dinámica.

Si se omite el tipo de datos por defecto es (como siempre) del tipo variant.

Nota: Al ejecutar la sentencia Dim todos los elementos de las matrices son inicializados a cero si el tipo de datos es numérico y a NULL si son caracteres.

3.2.4.2 Estructuras.

Sólo pueden aparecer en la sección de declaración de un módulo. Su modo de utilización es el siguiente:

Private Type *Ficha* Dim Nombre As String*15 Direccion As String*30 Telefono As Long ... **End Type**	Las sentencias **Type** y **End Type** limitan los datos de la estructura. ***Ficha*** es el nombre que recibe la nueva estructura de datos. Accedemos a los miembros Nombre, Dirección,etc. a través del operador (.).

La siguiente tabla indica dónde y con qué ámbito se pueden declarar tipos de datos definidos por el usuario, así como variables de esos tipos:

Procedimientos/Módulos	Puede crear una estructura como...	Las variables pueden ser...
Procedimientos	No puede	Sólo locales
Módulos estándar	Private o Public	Private o Public
Formularios	Sólo Private	Sólo Private
Módulos de clase	Sólo Private	Sólo Private

Se pueden generar matrices de estructuras del siguiente modo:

Dim Clientes() *As Ficha*

3.2.4.3 Objetos

Un objeto es una encapsulación da datos lógicamente relacionados entre sí y el código correspondiente para manipularlos. Los objetos de Visual Basic se crean a partir de clases. Podemos realizar las siguientes operaciones entre variables y objetos:

1. Asignación de un objeto a una variable objeto: Set varObj = RefObj

2. Múltiples ejemplares de un formulario. En una aplicación en la que existen varios formularios, se aplica la palabra clave **Me** que siempre referencia al ejemplar de formulario en el que el código se está ejecutando.

3.2.4.4 Colecciones

Visual Basic proporciona colecciones para poder acceder a todos los formularios de una aplicación o a todos sus controles.

Una colección contiene sólo los elementos activos (por ejemplo, los formularios cargados en un determinado momento). También existen colecciones de objetos, de controles y de objetos genéricos.

3.2.5 Procedimientos.

La programación con procedimientos tiene dos grandes ventajas: la primera es que permite dividir la utilización de la filosofía “divide y vencerás” en la creación del

código de una aplicación, y la segunda es que los procedimientos utilizados en una aplicación pueden servir a otra (reutilización del código).

En Visual Basic se usan varios tipos de procedimiento:

- Procedimientos **Sub** que no devuelven ningún valor
- Procedimientos **Function** que devuelven un valor
- Procedimientos **Property** que permiten crear propiedades para una clase.

Además estos pueden ser intrínsecos (los predefinidos en Visual Basic) o extrínsecos, definidos por el usuario.

3.2.5.1 Procedimientos para cadenas de caracteres.

Función / Sintáxis	Utilización/Ejemplo	
Len/*var* = **Len**(*exp*)	Da como resultado en ***var*** el número de caracteres de ***exp***. Por ejemplo: Exp = "Fran", Var = Len(exp) donde ' ***** var = 4 *****.	
Left/ *V* = **Left**(*exp*, *n*)	Devuelve los *n* caracteres más a la izquierda de *exp*	V = Left("Oso", 2) ' v = "SO"
Right/ *V* = **Right**(*exp*, *n*)	Idem a Left pero hacia la derecha.	
Mid/ *Var* = **Mid**("cad",n, [m])	Da como resultado una subcadena de una cadena de caracteres	
Instr/*Pos* = **Instr**([n], *c1*, *c2*)	Da como resultado la posición del primer carácter de la subcadena *c1* en *c2*. Ej: Print(10, "*")	
String/*Cad* = **String**(*n*,*carácter*)	Imprime *caracter n* veces en pantalla. Por ejemplo: Print(3, '*)	
Str/ *Var* = Str(número)	Devuelve en *var* el carácter equivalente del número.	
Val/*Num* = **Val**(*ExpCadena*)	Da el valor numérico de una cadena de caracteres.	
Chr/ *Var* = **Chr**(*Num*)	Da como resultado el carácter ANSI de *Num*.	
Asc/*Cd* = **Asc**(*Cad*)	Da el código de carácter *cd* del primer carácter de *cad*.	
Space/*Cad* = **Space**(*núm*)	Devuelve en *cad* una cadena de espacios = *num*.	
Date/Cad = **Date**	Devuelve en *cad* la fecha actual del sistema. Como ejemplo se cita: Cad = date '12/12/03 donde Date = "01/02/03"	
Time/Cad = **Time**	Idem a Date pero con la hora del sistema.	
Lcase/*CadMin* = **Lcase**(*Cad*)	Devuelve la cadena *cad* en minúsculas *CadMin*. Por ejemplo: Cmin = Lcase("AA") o 'Cmin = "aa"	
Ucase/*CadMay* = **UCase**(*Cad*)	Idem a Lcase con Mayúsculas.	
LTrim/Rtrim/Trim *Cad* = **Ltrim**(*Cadena*)	Copia en *Cad* la *Cadena* pero sin espacios en la derecha (Ltrim) o en la izquierda con (Rtrim)	
Now	Devuelve valor tipo Date que contiene fecha y hora.	

3.2.5.3 Procedimientos para expresiones numéricas.

Función Trigonométricas	**Utilización/Ejemplo**
Sin/Cos/Tan/Atn *valor* = **Sin**(*ángulo*)	Da como resultado en *valor* el seno del *ángulo*.
Funciones Logarítmica y Exponencial	**Utilización/Ejemplo**
Var = **Log**(*expresión*) *Var* = **Exp**(*expresión*)	• Log: Log neperiano en base e. • Exp: Inversa de Log.
Fix/ *Var* = **Fix**(*expresión*)	Da como resultado un entero que resulta de truncar el valor de una expresión numérica.
Otras funciones	**Utilización/Ejemplo**
Int/ *Var* = **Int**(*expresión*)	Da el mayor número entero que sea menor o igual que el argumento.
Abs/ *Valor* = **Abs**(*expresión*)	Da el valor absoluto de una *expresión*. Por ejemplo: Print abs(-19) que imprime '19
Sgn/ *Var* = **Sgn**(*expresión*)	Indica en *var* el signo de *expresión*. Si var = 1 exp es positiva, si var = -1 exp negativa y si var = 0 exp = 0.
Sqr/ *Var* = **Sqr**(*exp*)	Da en *Var* la raíz cuadrada de *exp*.
Timer/Var = **Timer**	Da el número de segundos transcurridos desde media noche.
Rnd/ *Var* = **Rnd**(*expr*)	Devuelve un número aleatorio del tipo **single** comprendido entre 0 y 1. Ej *a = Rnd()*
Randomize/Randomize[n]	Activa el generador de números aleatorios a partir de un número determinado. [n] es una expresión entera.

3.2.5.4 Funciones definidas por el usuario

La base de una aplicación de VB la forman sus procedimientos. Un procedimiento concluido por un evento es el código que se ejecuta cuando un objeto reconoce que se ha producido un determinado evento.

3.2.5.4.1 Ámbito del procedimiento:

Por omisión es Público pero puede cambiarse a voluntad del usuario.

3.2.5.4.2 Crear un procedimiento:

Haga clic en la ventana *ver código* y luego "*agregar procedimiento...*" del menú herramientas. También se puede rescribir como **Sub** o **Function** en la ventana de código. Al pulsar Enter, VB completará el esqueleto.

La siguiente tabla muestra la sintaxis típica de los tipos de procedimientos de usuario (*Function* y *Sub*)

Funciones (Function)	**Procedimientos Sub**
[Static] [Private] Function Nombre + + ([Parámetros]) [As tipo] Sentencia(s) Nombre = expresión Exit Function Sentencia(s) Nombre = expresión End Function	[Static] [Private] Sub Nombre [Parámetros]) Sentencia(s) Exit sub Sentencia(s) End sub 'Los Sub NO devuelven ningún valor!

Nota: La diferencia entre Function y Sub es únicamente que Sub no devuelve ninguna expresión.

Existen las siguientes variantes de procedimientos:

1. **Recursivos**. los procedimientos se llaman a sí mismos
2. **Con argumentos opcionales**: se pueden incluir o no en la invocación
3. **Con número indeterminado de parámetros**.

Los siguientes ejemplos clarifican el funcionamiento de cada caso.

1. Recursivos.	**2. Argumentos Opcionales**
Function Factorial (N As Integer) As Long Sentencia(s) Factorial = N*Factorial(N-1) Sentencia(s) End Function	Private Sub miProceso(Optional n=3 As Integer) **3. Número indeterminado de parámetros.** Public Function Maxim(ParamArray num()) ' Hay que especificarlo con ParamArray.

3.2.6 Cajas de diálogo

Cuando queremos introducir datos en un momento dado es una buena opción utilizar cajas de diálogo que interactúan directamente con el usuario. Hay básicamente tres tipos de cajas de diálogo: predefinidas (InputBox y MsgBox), personalizadas y comunes (Abrir,Imprimir...)

3.2.6.1 Cajas de diálogo predefinidas.

Como ya se introdujo, son ***MsgBox*** e ***InputBox***. Veamos primero ***MsgBox***.

La función o sentencia *MsgBox*, posee la siguiente sintaxis:

ValRetorno = **MsgBox**(*mensaje*, [,botones][,título])

- **Mensaje** es la cadena con a la que el usuario responderá
- **Botones** indican los tipos de botones que aparecerán en la ventana y pueden ser los siguientes:

vbOkCancel	VbOkOnly	vbQuestion	vbCritical	VbInformation
vbExclamation	VbYes	vbNo	vbYesNo	vbYesNoCancel

- **Título** es el título que posee la caja de diálogo mostrada.
- **ValRetorno** indica el botón que se ha pulsado y éste puede ser:

1 = Aceptar	2 = Cancelar	3 = anular	4 = Reintentar
5 = Ignorar	6 = Sí	7 = No	

Un ejemplo de uso sería el siguiente:

Resp = **MsgBox**("¿Has desayunado?", vbYesNo+vbQuestion, "Caja Diálogo")

3.2.6.2 Cajas de diálogo personalizadas.

Obligan al usuario a introducir una palabra de paso. VB proporciona dos posibilidades: **PasswordChar** y **MaxLength,** que facilitan la manipulación de la palabra introducida por el usuario. La propiedad **PasswordChar** especifica el carácter que será visualizado (por ejemplo, asteriscos) y **MaxLength** la longitud máxima de la palabra (por ejemplo 5 dígitos).

3.2.6.3 Casillas de verificación: lo más importante a estudiar de estos elementos es su propiedad *value*.

3.2.6.4 Botón de opción: Idem al punto anterior ya que cada botón es independiente del resto.

3.2.6.5 Macros o caja de grupo: Permiten identificar fácilmente a un grupo de controles. Poseen propiedades propias (título, color...). Para crear un macro se debe ir a *Herramientas→Frame*. Para introducir las opciones deseadas se arrastran botones de opción o casillas de verificación al interior del Marco creado.

3.2.6.6 Listas y Listas desplegables: Poseen varias funcionalidades básicas.

1. **AddItem:** Para añadir elementos a una lista o lista desplegable y su sintaxis es:

   NombreLista.**AddItem** *elemento* [,*índice*]

   donde *elemento* es el nombre del nuevo elemento a insertar en la posición marcada por *índice*.

2. **RemoveItem:** Elimina un elemento de la lista o lista desplegable y su sintaxis es: NombreLista.**RemoveItem** *índice*

3. **ListCount:** Da como resultado el número de elementos que hay en la lista. Su sintaxis es [formulario.][NombreLista.]**ListCount**. También son interesantes las propiedades ListIndex.

3.2.6.7 Barras de desplazamiento: Son utilizadas para desplazar la información de arriba abajo o de derecha a izquierda. Sus propiedades más importantes son Max, Min y Value.

3.2.6.8 Colores: Se puede cambiar el color de los objetos mediante las propiedades BackColor y ForeColor o mediante las funciones RGB o QBColor cuya sintaxis es:

RGB(*rojo*, *verde*, *azul*)

donde rojo, verde y azul son enteros comprendidos entre 0 y 255 y especifican el nivel de intensidad del color respectivo.

3.2.6.9 Cajas de diálogo comunes.

El control *Microsoft Common Dialog* permite visualizar las cajas de diálogo más comunes de las aplicaciones de Windows. Para introducir dicho control en la barra de herramientas se debe seguir la siguiente secuencia de pasos; *Proyecto→Componentes→CommonDialogControl*.

Este control permite realizar las siguientes operaciones: Abrir, Guardar, Guardar Como, Imprimir, Fuente y Color. Para ello deben invocarse los métodos **showOpen**, **ShowSave**, **ShowColor**, **ShowFont**, **ShowPrinter** y **ShowHelp** respectivamente.

3.2.6.10 Control de un sistema de ficheros.

VB proporciona herramientas para operar con archivos del modo que se expuso en el punto anterior, pero quizás de un modo más tedioso. Para ello estúdiense las propiedades *DriveListBox*, *DirListBox* y *FileListBox* que muestran la unidad de disco, directorio y fichero respectivamente.

3.2.7. Ficheros de datos.

Hasta ahora todas las aplicaciones obtenían los datos a través del usuario; sin embargo, en la mayor parte de las aplicaciones se requiere almacenar los datos obtenidos para cargarlos posteriormente o viceversa. Es por esto que aprender a trabajar con ficheros de datos resulta de vital importancia en el mundo de la programación.

3.2.7.1 Sentencias básicas sobre ficheros

Sentencia	Utilidad	Sintaxis/Ejemplo
Kill	Elimina cualquier tipo de fichero almacenado en el directorio actual.	Kill "PathName\File" Kill "NombreFichero.Dat"
Name	Renombra y mueve ficheros.	Name "File" As "File2" Name "Prova.txt" As "c:\otroDir\Prueba.dat"
ChDir	Cambia el directorio actual.	ChDir "NewDirectory" ChDir "c:\Datos"

3.2.7.2 Abriendo ficheros para el acceso de información

- Sentencia **Open**. Da acceso para leer/escribir ficheros. Su sintaxis es:

Open "*Nombre*" ***For*** Modo ***As*** *NúmeroFichero*

donde "*Nombre*" es la ruta y nombre del fichero: "C:\Pruebas\Hola.txt", Modo indica la finalidad para la que se abre el fichero indicado en "Nombre" y puede ser:

Output: Para escribir en el fichero secuencialmente. Crea el fichero cada vez luego si el fichero ya existe se destruirá para crearse de nuevo.

Append: Abre el fichero para añadir los datos de modo secuencial a los que ya existen; por lo tanto no destruye datos, sólo añade otros.

Input: Abre un fichero para leerlo desde el principio. Si el fichero no existe se generará un error.

NúmeroFichero es una expresión entera entre 1 y 511 asociado al fichero abierto. Este número será el que identifique al fichero en las operaciones E/S.

- Sentencia **Print #**, permite grabar datos secuencialmente en un fichero:

  Print #NúmeroFichero, expresiones[{,/:}]

  donde #NúmeroFichero ya se ha presentado anteriormente y las expresiones son una lista de expresiones numérico y/o de caracteres a ser escritas sobre el fichero. Un ejemplo de uso sería: Print #NúmeroFichero, "David".

- Sentencia **Write #**, idéntica a print # pero permite insertar comas entre cadenas y es más flexible en general. Su sintaxis es:

  Write #NúmeroFichero, expresiones[{,/:}]

  y un ejemplo de uso sería: Write #1, "Hola", " ", "Adios", ",", "Feo"

- Sentencia **Close**, finaliza las operaciones de E/S del fichero abierto con Open, su sintaxis es la siguiente:

Close #NúmeroFichero, #NúmeroFichero2, ...

donde en una línea podemos cerrar todos los ficheros que queramos. Un ejemplo de uso es: Close #1, #2, #3

- Sentencia **Input**, lee datos procedentes de un fichero secuencial y los asigna a las variables especificadas. Su sintaxis es la siguiente:

  Input #NúmeroFichero, variable, [,variable],...

  donde *variable* es el nombre de una variable numérica o de caracteres que recibe el dato a almacenar. Un ejemplo sería: Input #1, Nombre.

- Sentencia **Line Input** #, lee una línea de un fichero secuencial en disco ignorando los delimitadores y la asigna a una cadena de caracteres.

  Line Input #Fichero, variable

- Función EOF, indica si hemos llegado al final de un fichero. Su sintaxis es:

  v = EOF(NúmerFichero)

  Devuelve *true* cuando ha llegado al final, *false* en caso contrario.

3.2.7.3 Modelo de objetos del sistema de ficheros.

A partir de la versión 6 de VB se aparece una nueva forma de trabajar con ficheros fundamentada en el modelo de objetos del sistemas de ficheros (OSF) que facilita las operaciones básicas. Este modelo dispone de los objetos:

Objeto	Descripción
Drive	Obtiene información acerca de las unidades del sistema, espacio disponible,...
Fólder	Permite crear, eliminar, mover y obtener información de carpetas.
File	Idem a fólder pero para archivos.
TextStream	Permite leer y escribir ficheros de texto.
FileSystemObject	Es el objeto principal, proporciona métodos para crear, eliminar u obtener información. También para controlar unidades, carpetas y ficheros.

3.2.8. Acceso a una base de datos

Se pretende dar a conocer en este apartado las ventajas de trabajar con Bases de datos (BD) en lugar de hacerlo con ficheros. Se define una base de datos como un *conjunto de datos clasificados y estructurados guardados en uno o varios ficheros pero referenciados como si de un único fichero se tratara*.

VB incluye un administrador visual de datos que permite crear bases de datos Microsoft .Jet. También puede utilizar *Microsoft Access*.

Los datos de una base de datos se clasifican en *campos* (Nombre, apellidos,...) y *registros*. A su vez una BD puede contener más de una tabla, que es una agrupación de muchos registros, siendo un registro la información correspondiente de un campo.

3.2.8.1 El control de datos ADO.

Permite crear aplicaciones para visualizar, editar y actualizar información de muchos tipos de BD como *Microsoft Access*, SQL, Oracle, etc. y en general de cualquier origen escrito en OLE DB. Para comunicar una aplicación con una BD a través de ADO hay que seguir dos pasos:

1. Añadir el control al formulario y especificar la BD a trabajar. Esto permite editar, ver y actualizar registros de la BD.

2. Añadir otros controles y enlazarlos al control de datos ADO para visualizar información de cada uno de los registros de la base. Destacar aquí el uso de controles como *CheckBox, ListBox, PictureBox y TextBox* y controles ActiveX como *DataGrid, DataCombo, Chart y DataList*.

3.2.8.3 Utilización del control ADO.

Para la correcta utilización del control ADO es necesario seguir la siguiente secuencia de acciones:

1. Crear una BD con *Microsoft Access* o bien con VisData que es una aplicación que incluye VB en el menú *Complementos→ Administrados Visual de Datos* (VisData).

2. Crear un vínculo de datos OLE DB.

3. Diseñar el formulario.

4. Vincular el control de datos.

5. Programar el control de datos.

Crear un vínculo de datos OLE DB: Se debe crear un origen de datos OLE DB para cada BD a la que se desea tener acceso a través de ficheros de enlace de datos con el usuario "*.udl*".

Vincular el control de datos, para tal proceso, el principal paso es poner el control ADO en el formulario. Si no está en la barra de herramientas, búsquelo en *Proyecto→ Componentes→ Microsoft ADO DataControl*.

Programar el control de datos: Si no queremos que se vea nuestro control de datos proporcionado por VB, podemos ocultarlo y programar la misma funcionalidad con botones mediante el siguiente código:

- Botón Ir al **primer** registro: *Adodc1.Recordset.MoveFirst*
- Botón Ir al registro **anterior**: *Adodc1.Recordset.MovePrevious*
- Botón Ir al registro **siguiente**: *Adodc1.Recordset.Movenext*
- Botón Ir al **último** registro: *Adodc1.Recordset.MoveFirst*

Si queremos que estando en el último registro y pulsamos siguiente se pase al primer registro de la BD, programamos las siguientes líneas:

```
Private Sub Siguiente_Click()
   Adodc1.Recordset.MoveNext
   If Adodc1.Recordset.EOF then
      Adodc1.recordset.MoveLast
   End If
End Sub
```

```
Private Sub Anterior_Click()
   Adodc1.Recordset.MovePrevious
   If Adodc1.Recordset.BOF then
      Adodc1.recordset.MoveFirst
   End If
End Sub
```

El código de la derecha pertenece al caso contrario del mencionado.

3.2.9. Trabajando con menús.

Un menú es una forma de proporcionar al usuario un conjunto de órdenes lógicamente relacionadas y agrupadas bajo un mismo título. Los elementos que pueden componer un menú son *órdenes*, *submenús* y *separadores*.

Por convenio una orden seguida de "..." indica que se abre una caja de diálogo. Cuando se hace clic en un submenú se despliega una nueva lista de elementos. Un

separador de menús sirve para separar grupos de órdenes distintas. Por ejemplo para discernir entre las operaciones de ***abrir***, ***guardar***, ***guardar como***, etc de las operaciones como ***salir,*** etc propias del menú *archivo*.

Para crear menús se debe pulsar ctrl+E o en *Herramientas* → *editor de menús*. Para crear un separador se introduce en el campo *caption* un único guión (-). El carácter "&" sirve para subrayar una letra de una orden (por ejemplo Archivo) para indicar la tecla de acceso rápido a dicha orden. También se pueden definir *shortcuts* y otras opciones típicas de los menús de Windows.

El manejo del editor de menús de VB es muy sencillo e intuitivo, así que el mejor modo de aprender es probando.

Matrices de controles: Una matriz de controles es un conjunto de controles que comparten el mismo nombre y tipo, por lo cual también comparten procedimientos.

3.2.10 Controles ActiveX

3.2.10.1 Introducción

Los ficheros activeX son ficheros independientes con extensión ".ocx" que añaden funcionalidades a VB. Controles de este tipo son los ADO vistos anteriormente, barras de herramientas, barra de progreso, etc. Los controles ActiveX se encuentran en el fichero *mscomctl.ocx* (*Microsoft Windows Common Controls*).

VB puede incluir componentes ActiveX como aplicaciones Word, Excel o Access, que proporcionan a VB objetos para ser manipulados debidamente. Añadir un control ActiveX a nuestra aplicación en VB aumenta la flexibilidad de programación

ya que se introducen nuevos eventos, métodos y propiedades. Pueden encontrarse controles del tipo ActiveX desarrollados por otros programadores, pero puede que no sean compatibles con todas las versiones de VB. Los proporcionados por Windows son de 32 bits.

3.2.10.2 Ejemplo de control ActiveX, el control cuadrícula

Sirve para visualizar información distribuida en filas y columnas. La información se visualiza en celdas que son intersecciones de una fila con una columna. El ratón puede actuar sobre las dimensiones de las celdas si ha establecido la propiedad **AllowUserResizing**. El control **MSHFlexGrid** proporciona funcionalidades para operar con las celdas como si de una hoja de cálculo se tratara, ordenar, fusionar,...

Toda la información presentada hasta el momento ha sido extraída tanto de libros de programación orientada a objetos [9] y [10] genéricos como de otros específicos al lenguaje de Visual Basic como [3], [4], [5] y [6].

3.3 El programa *Calcicat*.

Como se mencionó al comienzo del capítulo, *Calcicat* está compuesto por dos bloques fundamentales: los formularios y los módulos.

Un formulario es lo que típicamente interacciona con el usuario, es decir, son las ventanas propias de cualquier programa donde se encuentran campos de verificación, texto, imágenes, etc.

Por otro lado, los módulos representan una parte más compleja ya que son secciones de código asociado a eventos, funciones, etc. y consecuentemente, representan la parte compleja de la programación aunque, eso sí, no la más importante, ya que un buen diseño y un programa interactivo es crucial para el éxito de la aplicación. La figura 3.1 esquematiza el citado concepto.

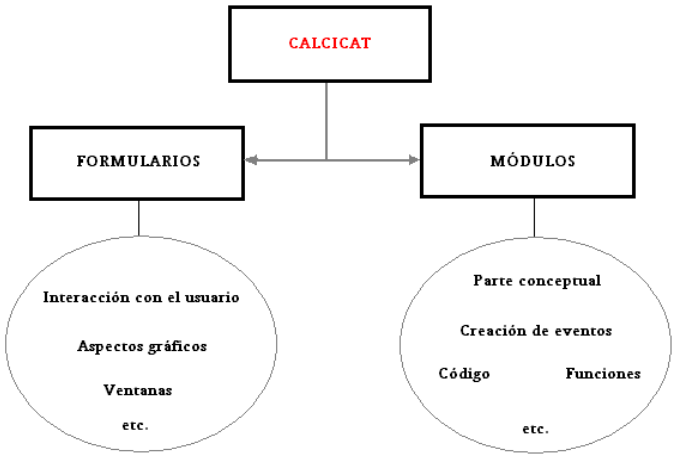

Figura 3.*1. Esquema conceptual de la composición de CALCICAT v1.0*

A continuación se pasará al estudio detallado de cada uno de estos bloques comenzando en primer lugar por la parte de diseño que comprende a los formularios, continuando con los módulos donde se presentarán las funciones que finalmente serán utilizadas en la programación por eventos que se asociará a cada uno de los formularios presentados en el siguiente punto.

Dicho orden de estudio es consecuencia directa de que el diseño del formulario no conlleva código asociado en cambio los módulos sí. Por otro lado en los módulos se declaran funciones que se utilizan posteriormente en la programación por eventos asociada a cada uno de los diseños de formulario presentados. Por ello, el capítulo presenta el citado orden, ya que se irán aprendiendo funciones según aparezcan en el código de manera secuencial.

3.3.1 Los formularios.

Como ya se ha anticipado en la introducción, los formularios constituyen una de las partes más importantes en la programación de un lenguaje visual pues de ellos se deriva el aspecto de las ventanas, la interactividad con el usuario, etc.

Calcicat es una aplicación que abarca en su interior un total de 30 formularios cuya extensión asociada es el formato "*.frm*" tal y como muestra la figura 3.2:

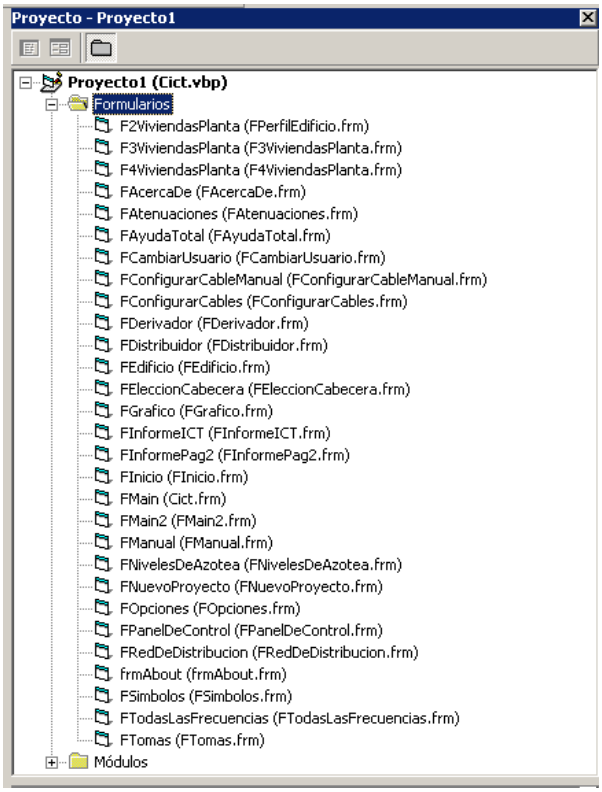

Figura 3.2. *Ventana de formularios de la aplicación Calcicat*

En toda aplicación es habitual disponer de un formulario principal que suele constituir el cuerpo de la aplicación; en el caso de *Calcicat* dicho formulario es "*Fmain.frm*" cuyo aspecto al hacer doble clic sobre él en la figura 3.2,se muestra en la figura 3.3.

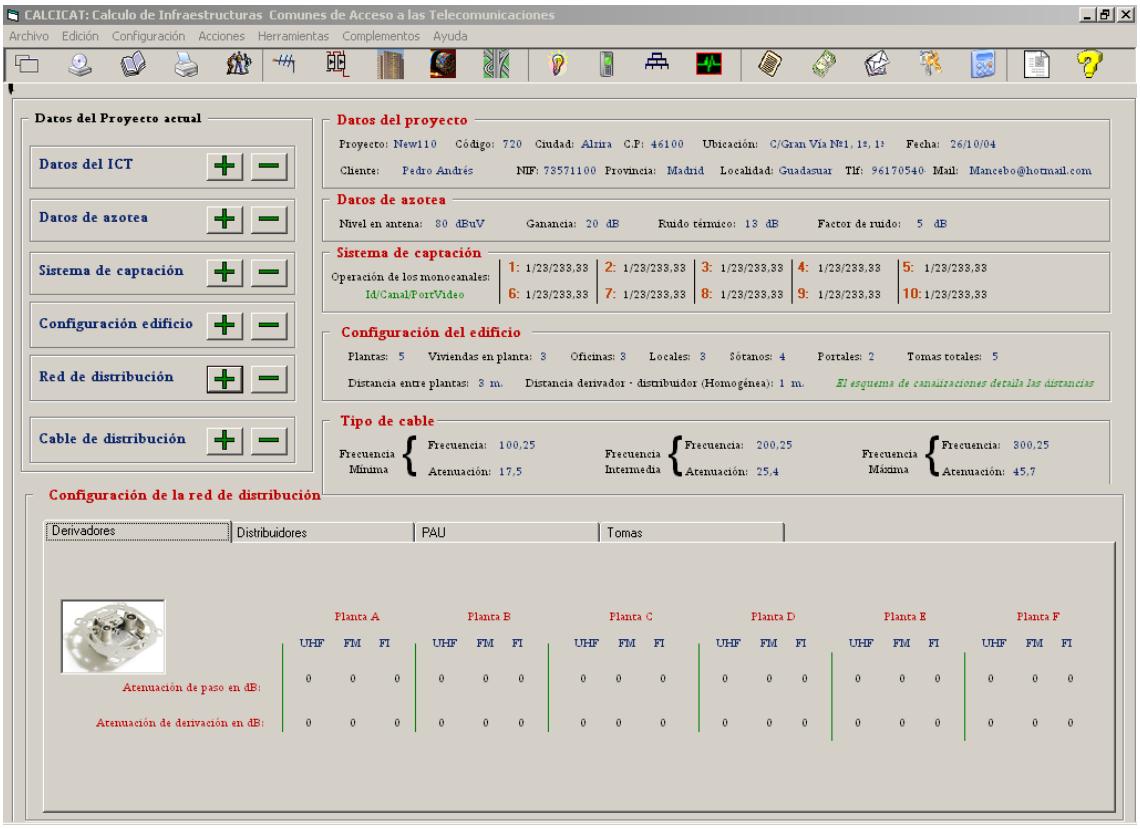

Figura 3.3. *Aspecto del formulario Fmain.frm*

Por cuestiones de nomenclatura y quizás de estandarización, según se vio en el punto de aprendizaje de VB, cada formulario tiene un nombre propio e identificativo de su finalidad precedido de la letra F indicando que se trata de un formulario. En el caso anterior era ***Fmain.frm*** donde se indicaba que se trataba de un formulario (F) principal (Main) de la aplicación ejecutada.

Si se hace doble clic sobre el formulario en tiempo de diseño aparecerá el código asociado a la aplicación. La figura 3.4 muestra parte del código de *Fmain.frm*.

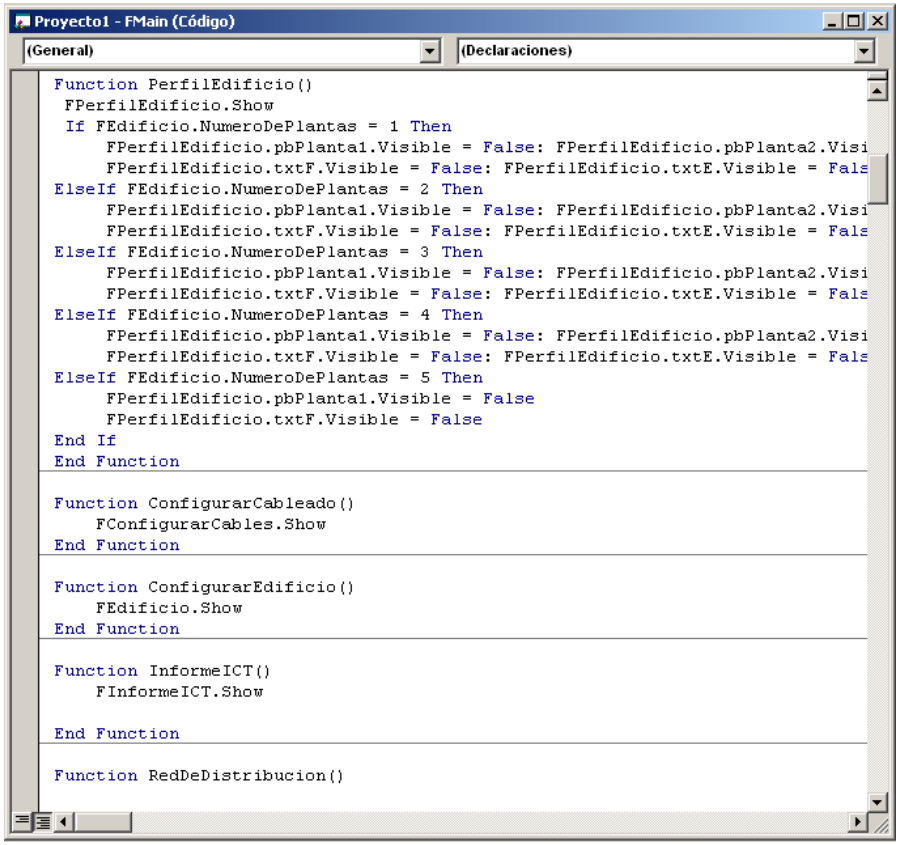

Figura 3.4. *Ventana de código asociada al formulario "Fmain.frm".*

Siguiendo con los formularios mostrados en la figura 3.2, la finalidad de "*F2ViviendasPlanta.frm*", "*F3ViviendasPlanta.frm*" y "*F4ViviendasPlanta.frm*" es, como su nombre indica, representar en pantalla un edificio de alturas comprendidas entre 1 y 6 plantas de 2, 3 o 4 viviendas en planta respectivamente. La figura 3.5 muestra el diseño asociado al formulario de 4 viviendas por planta (*F4ViviendasPlanta.frm*).

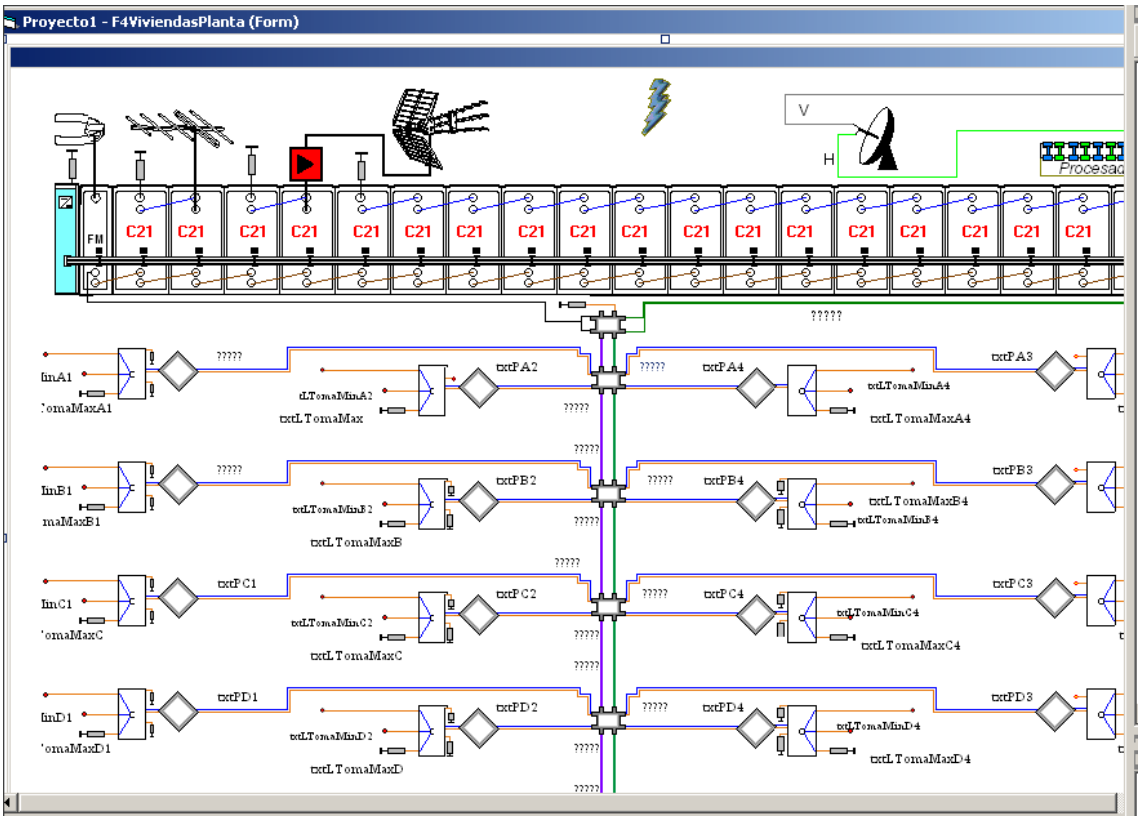

Figura 3.5. *Aspecto asociado al formulario "F4ViviendasPlanta.frm"*

Otros formularios de interés son los encargados de ir configurando paso a paso el proyecto del ICT. Para tal efecto se dispone de: "FConfigurarCable.frm", "FConfigurarCableManual.frm", "FDerivador.frm", "FDistribuidor.frm", "FEdficio.frm", "FElecciónCabecera.frm", "FNivelesDeAzotea.frm", "FNuevoProyecto.frm", "RedDeDistribucion.frm" y "FTomas.frm" cuyos diseños se muestran en las figuras 3.6 – 3.15.

- FconfigurarCable.frm

Figura 3.6. *Ventana de diseño del formulario "FconfigurarCable.frm"*

- FConfigurarCableManual.frm

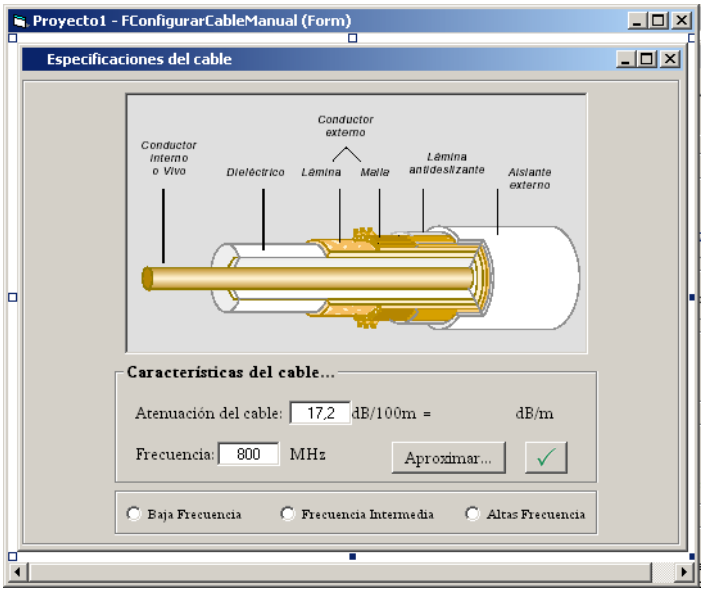

Figura 3.7. *Ventana de diseño del formulario "FconfigurarCableManual.frm"*

- FDerivador.frm

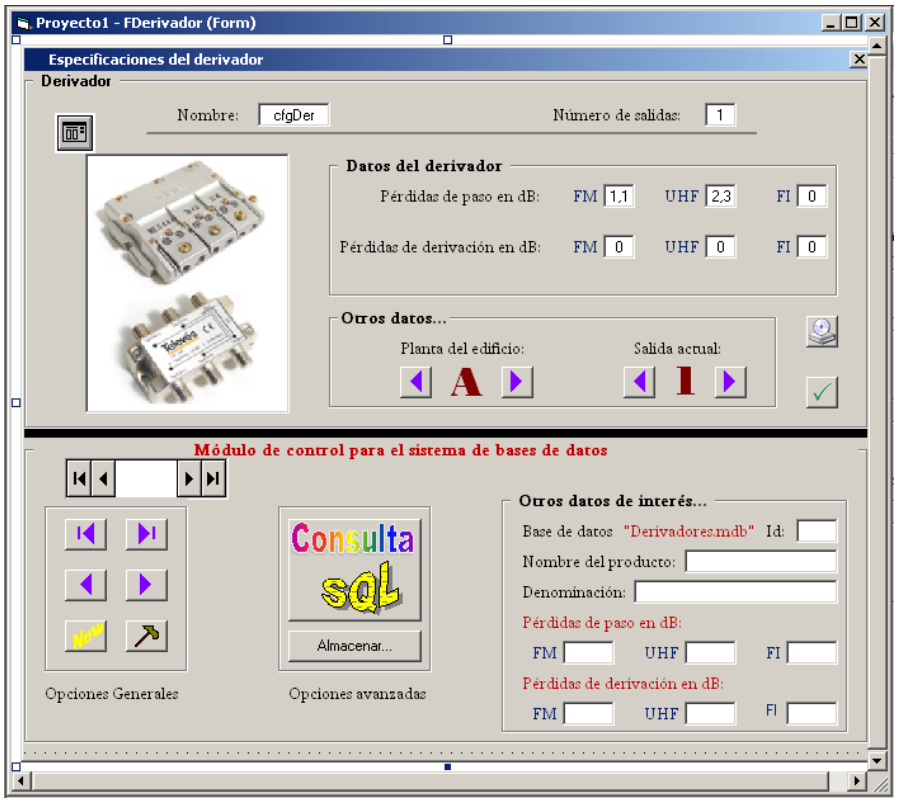

Figura 3.8. *Ventana de diseño del formulario "FDerivador.frm"*

- FDistribuidor.frm

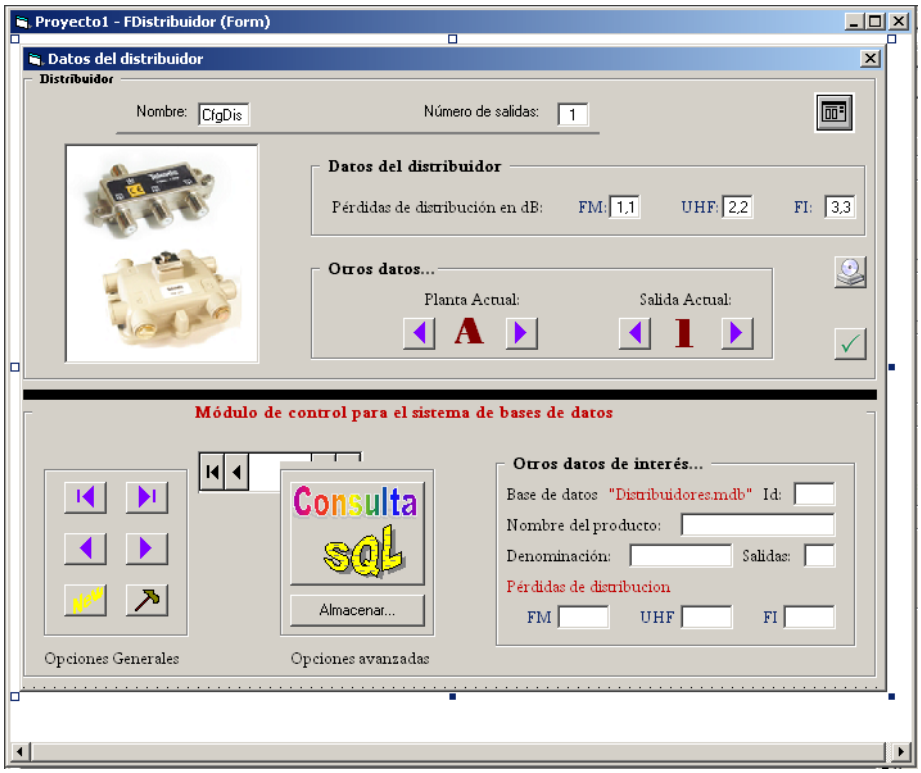

Figura 3.9. *Ventana de diseño del formulario "FDistribuidor.frm"*

- FEdficio.frm

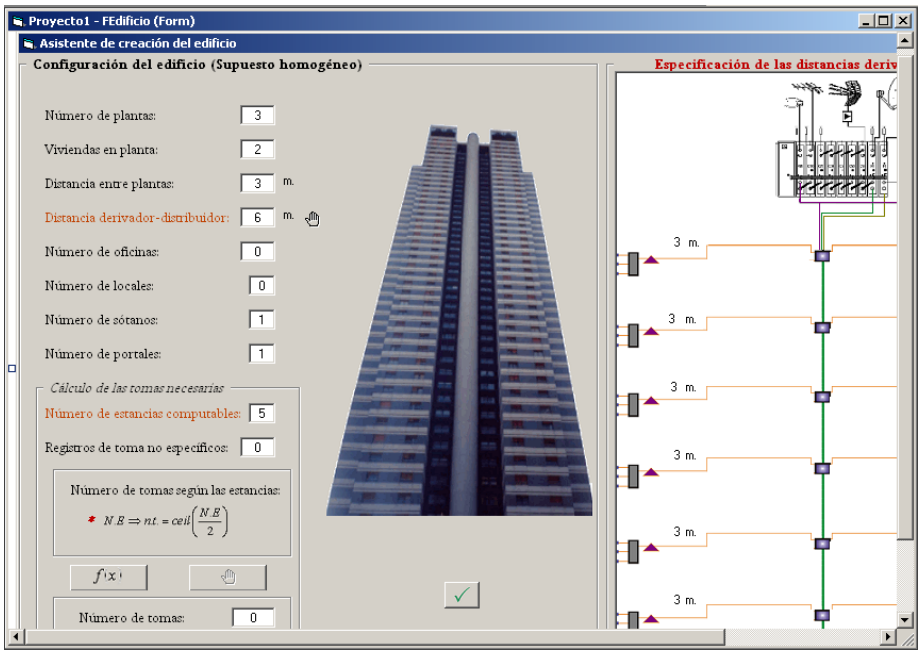

Figura 3.10. *Ventana de diseño del formulario "FEdificio.frm"*

- FElecciónCabecera.frm

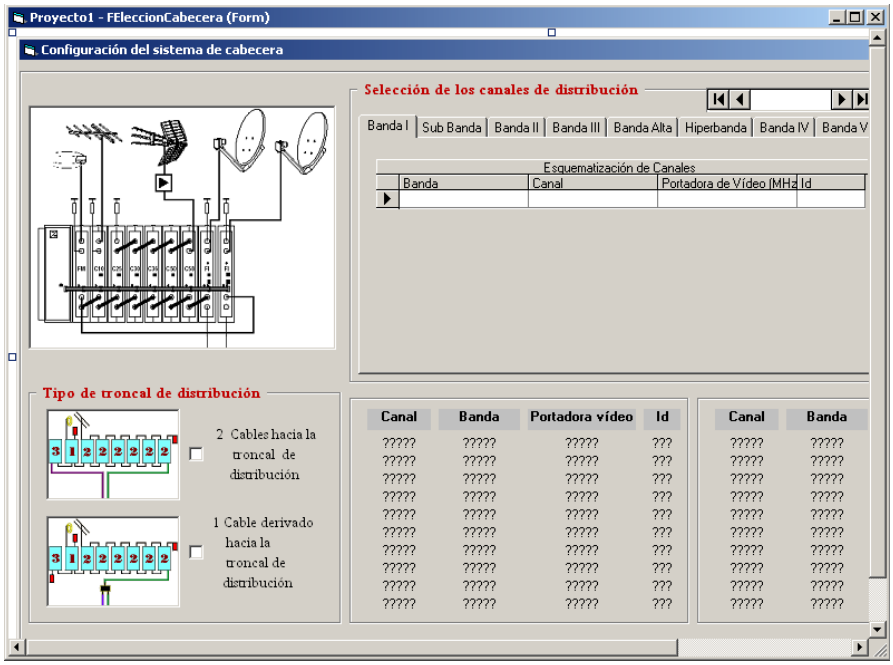

Figura 3.11. *Ventana de diseño del formulario "FElecciónCabecera.frm"*

- FNivelesDeAzotea.frm

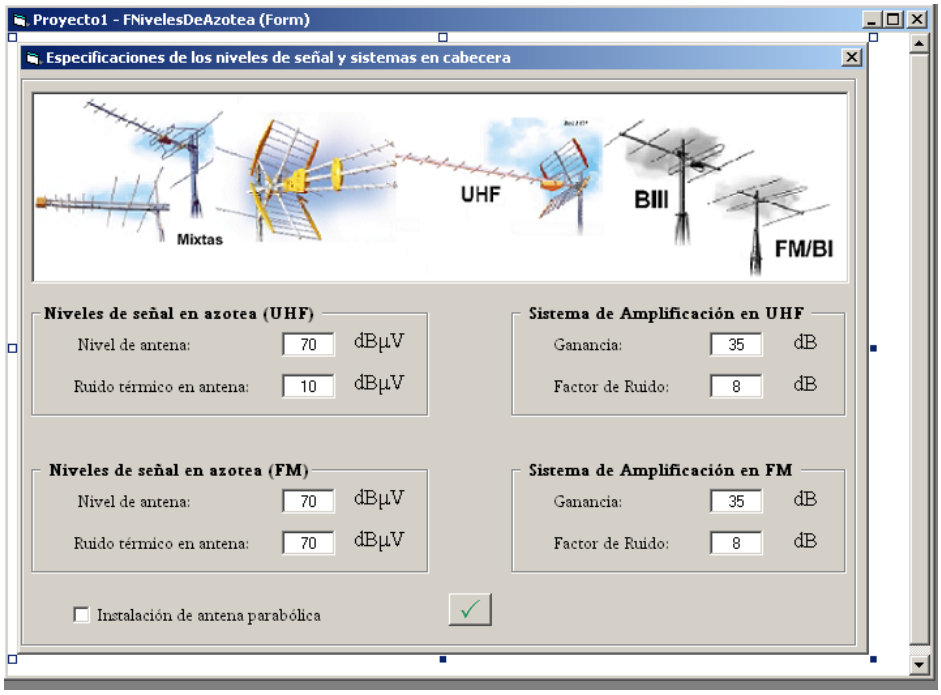

Figura 3.12. *Ventana de diseño del formulario "FNivelesDeAzotea.frm"*

- FNuevoProyecto.frm

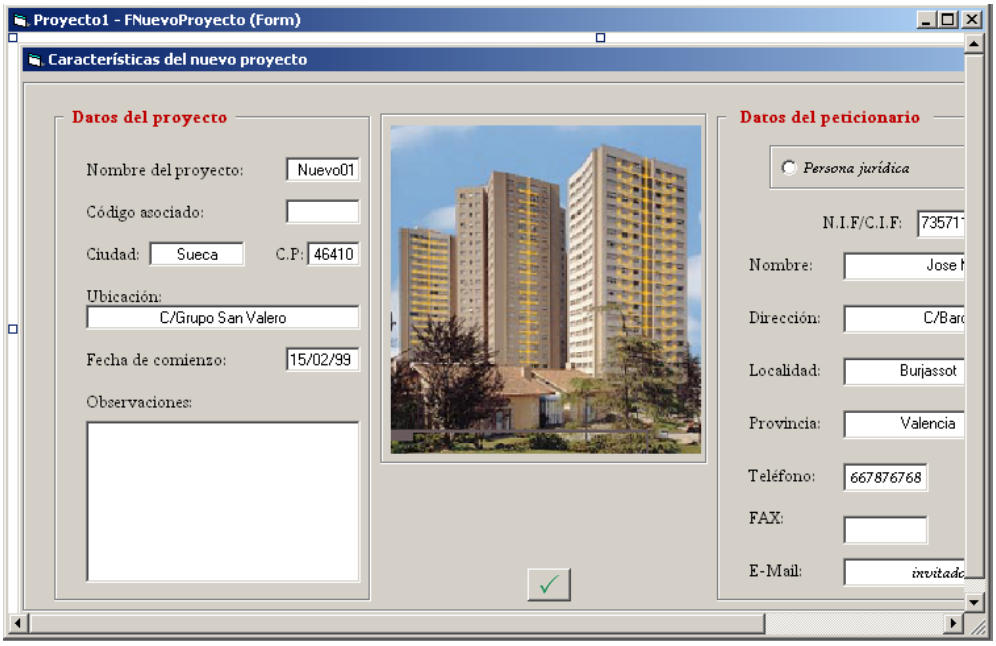

Figura 3.13. *Ventana de diseño del formulario "FNuevoProyecto.frm"*

- "RedDeDistribucion.frm"

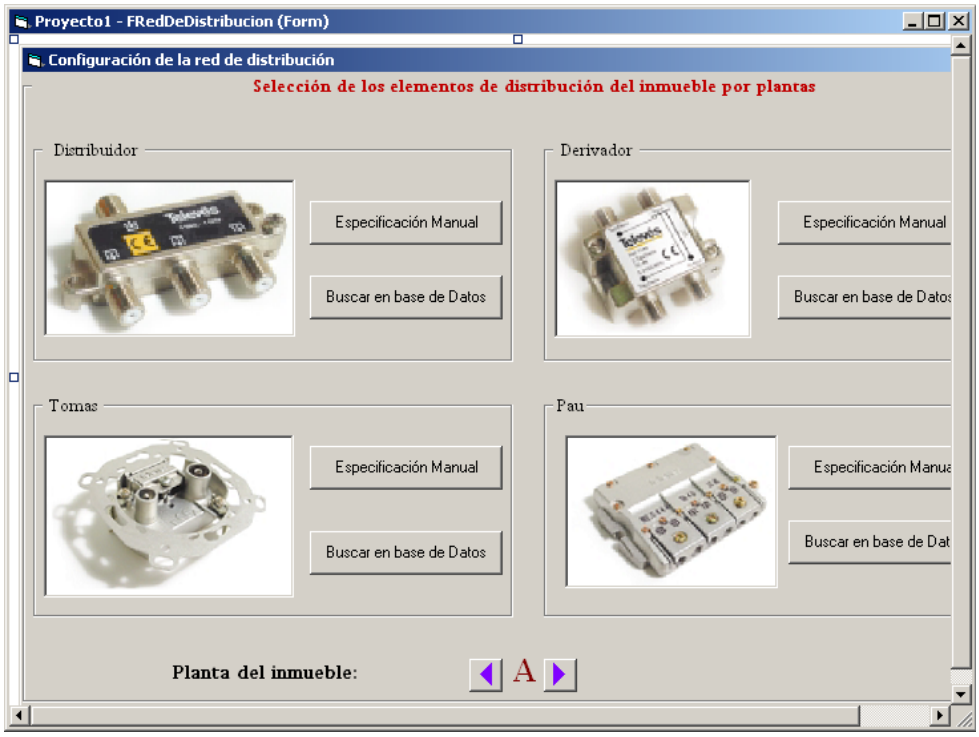

Figura 3.14. *Ventana de diseño del formulario "FredDeDistribucion.frm"*

- "FTomas.frm"

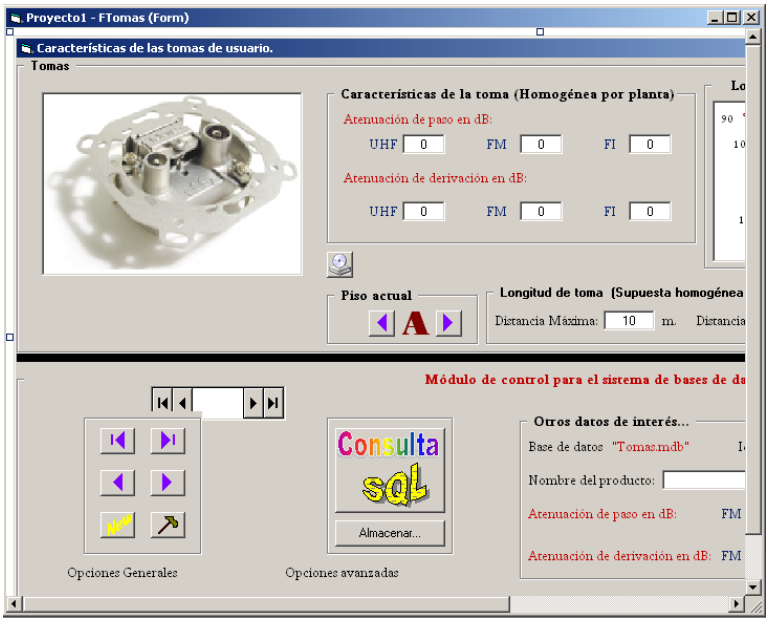

Figura 3.15. *Formulario "FTomas.frm"*

Otros de los formularios importantes son los encargados de operar una vez obtenido el cálculo del ICT, es decir el propio esquemático. Los formularios que corresponden a esta parte son: "FpanelDeControl.frm", "Fopciones.frm", "FSimbolos.frm", "FTodasLasFrecuencias.frm" y "FGraficas.frm" y se muestran en las figuras 3.16 – 3.20.

- "FPanelDeControl.frm"

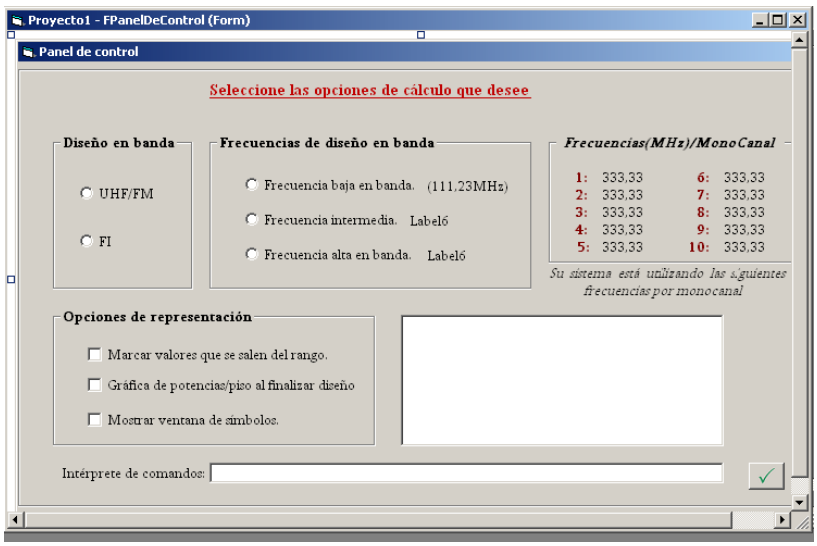

Figura 3.16. *Ventana de diseño del formulario "FpanelDeControl.frm"*

- "FOpciones.frm"

Figura 3.17. *Formulario "FOpciones.frm"*

- "FSimbolos.frm"

Fig 18. *Formulario "FSimbolos.frm"*

- "FTodasLasFrecuencias.frm"

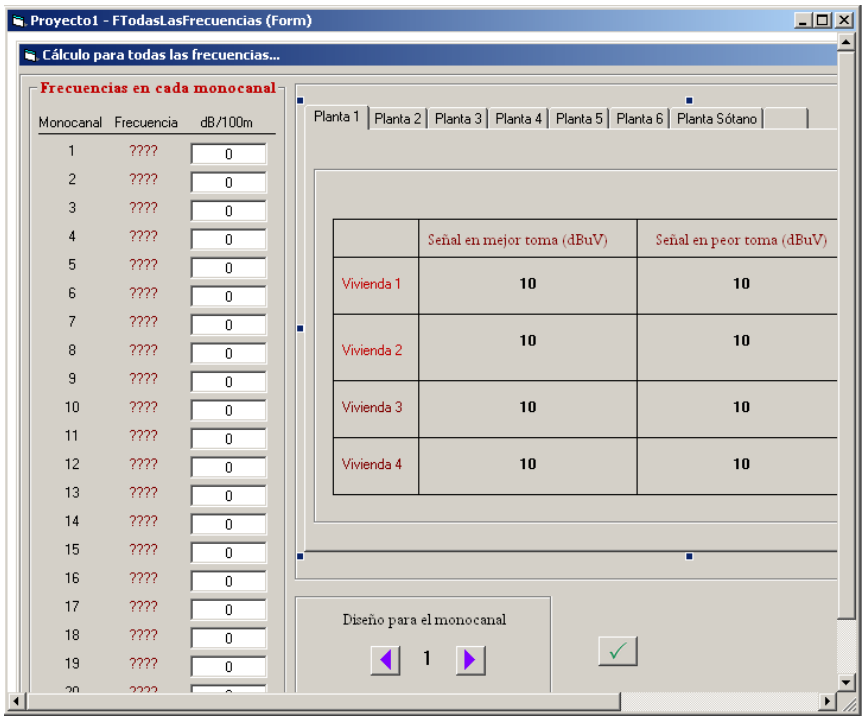

Figura 3.19. *Formulario "FTodasLasFrecuencias.frm"*

- "FGráficas.frm"

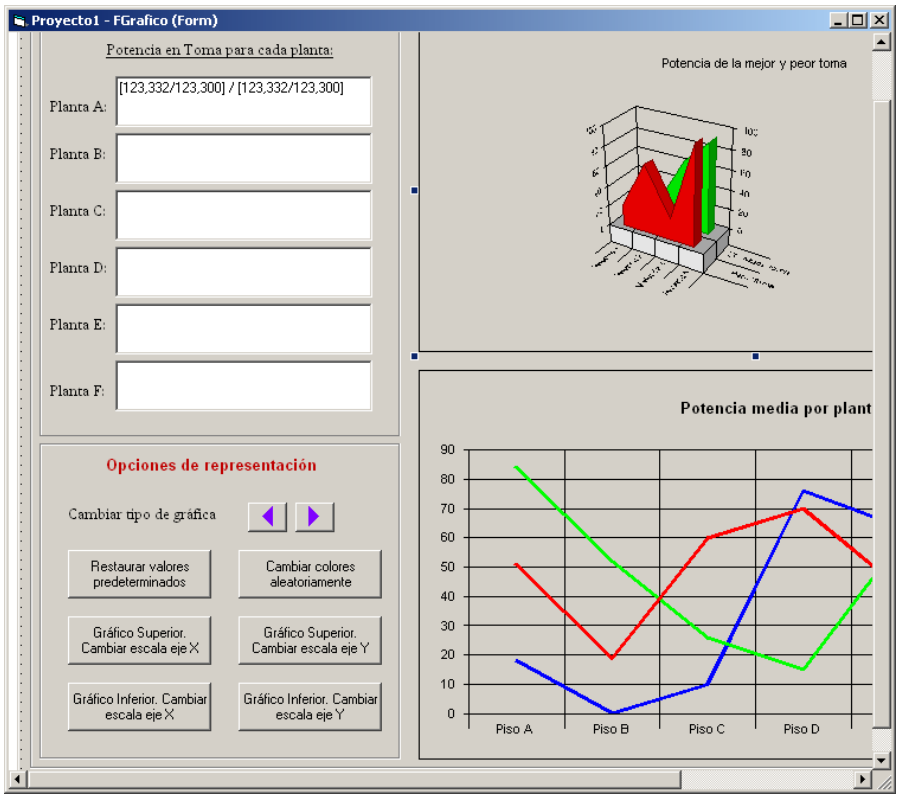

Figura 3.20. *Formulario "FGraficas.frm"*

Continuando con el análisis de la figura 3.2, aparecen formularios cuya misión es facilitar al máximo las tareas del diseñador del proyecto de ICT como, por ejemplo, el formulario encargado de generar el informe automático (memoria) cuyo nombre es "Finforme.frm" y su aspecto el mostrado en la siguiente figura 3.21:

- "FInforme.frm"

Figura 3.21. *Formulario "FInforme.frm"*

Además de los anteriores, se encuentran una serie de formularios cuya finalidad es puramente de ayuda al usuario como el "FacercaDe.frm" o el "FAyudaTotal.frm", mostrado en las figuras 3.22 y 3.23.

- "FAcercaDe.frm"

Figura 3.22. *Formulario "FAcercaDe.frm"*

- "FAyudaTotal.frm"

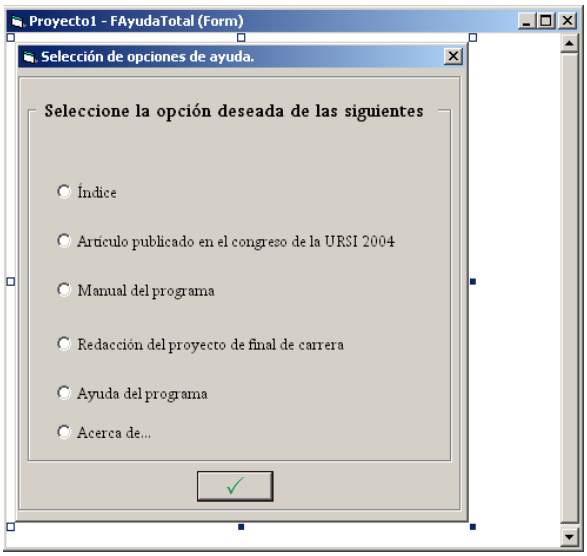

Figura 3.23. *Formulario "FAyudaTotal.frm"*

Finalmente el programa posee unos formularios encargados de identificar al usuario del software, autorizarle a utilizar sus proyectos previamente guardados, etc. Dicho formulario es "FcambiarUsuario.frm" y se muestra en la figura 3.24:

- "FCambiarUsuario.frm"

Figura 3.24. *Formulario "FcambiarDeUsuario.frm"*

Una vez vistos los formularios de la aplicación queda empezar con el estudio del código, si bien dicho estudio será sólo en términos generales ya que la totalidad del código representaría un único manual de más de 500 páginas de código puro. El apartado 3.3.2 tratará tanto los módulos del software como las funciones y secciones de código más relevantes del programa.

3.3.2 Los módulos

Como se ha visto en el punto anterior, VB ofrece la posibilidad de diseñar formularios de modo interactivo, en muy poco tiempo y con gran sencillez. Además de esto, también se ofrece la denominada "ocultación de información" en lo que al código se refiere. Ello provoca la obtención de programas claros y estructurados.

A partir de cualquier formulario de los vistos anteriormente, el usuario accede al código correspondiente con un simple "*doble clic*" sobre el objeto deseado. Por ejemplo, al hacer un doble clic sobre el objeto mostrado en la figura 3.25, aparecerá de inmediato su código asociado (evento) tal y como muestra la figura 3.26.

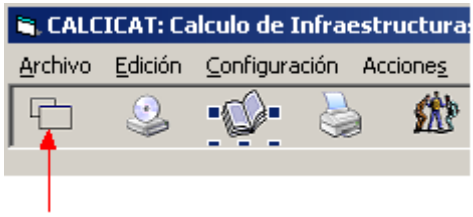

Figura 3.25. *Objeto cualquiera del formulario principal Fmain*

```
Function NuevoProyecto()

  Resp = MsgBox("Ésta opción cerrará el proyecto actual! ¿Desea continuar?", vbYesNoCancel + vbInformation, "Nuevo Proyecto.")

  If Respuesta = vbYes Then
     'El asistente guía al usuario en el desarrollo del nuevo proyecto de ICT
     FNuevoProyecto.Show
  ElseIf Respuesta = vbNo Or vbCancel Then
  End If

End Function
```

Figura 3.26. *Código asociado al objeto de la figura 3.25.*

Adicionalmente a este modo de programas objeto – evento, existe en VB la posibilidad de disponer de módulos (archivos con extensión .bas) que no son más que espacios en blanco donde el programador genera funciones, variables, etc. que se utilizarán en cualquier parte del programa. En este ámbito *Calcicat* dispone de un total de 11 módulos como muestra la figura 3.27:

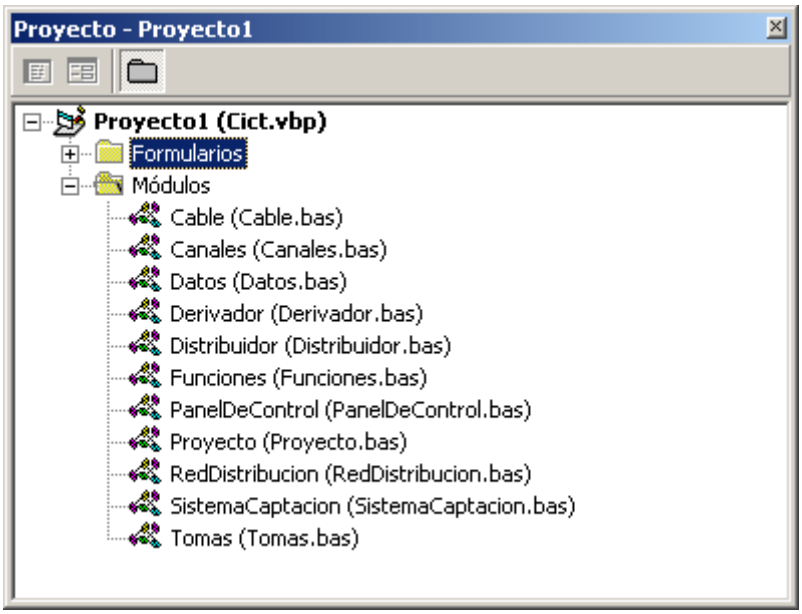

Figura 3.27. *Módulos utilizados en Calcicat*

Cada uno de los módulos mostrados tiene una finalidad bastante específica a excepción del módulo de datos "*Datos.bas*" y del módulo de funciones "*Funciones.bas*". Por ello, se estudiará en primer lugar los módulos específicos, esto es: "Cable.bas", "Canales.bas", "Derivador.bas", "Distribuidor.bas", "PanelDeControl.bas", "Proyecto.bas", "RedDeDistribucion.bas", "SistemaCaptacion.bas" y "Tomas.bas".

3.3.2.1 Módulos específicos

Como puede verse, el nombre que identifica a cada módulo da una idea bastante próxima a la tarea a desempeñar por éste, si bien en ocasiones extienden sus funcionalidades, pero esto no se tratará en este capítulo por no extender en exceso los contenidos. Dicho esto, se pasa a detallar cada uno de los módulos.

- "*Cable.bas*" y *"Canales.bas":* Ambos módulos son similares y en su interior solamente disponen de una función encargada de validar los datos insertados en el formulario correspondiente. Además se encargan de la impresión sobre

el panel de la aplicación principal y de la declaración de variables correspondientes. La figura 3.28 muestra una sección de código del módulo *"Cable.bas"*.

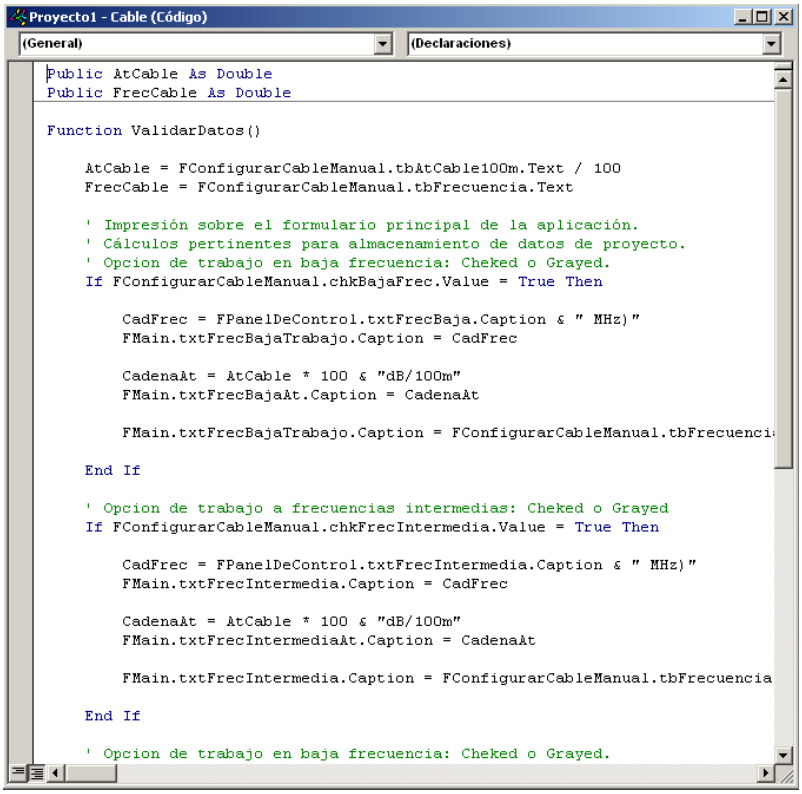

Figura 3.28. *Módulos "Cable.bas"*

- ***"Derivador.bas":*** La finalidad de este módulo se divide en tres funciones básicas siendo la primera de ellas la declaración de todas las variables de acceso público de las distintas plantas del inmueble en lo que al derivador se refiere. Dicho hecho queda constatado a la vista del código de la figura 3.29.

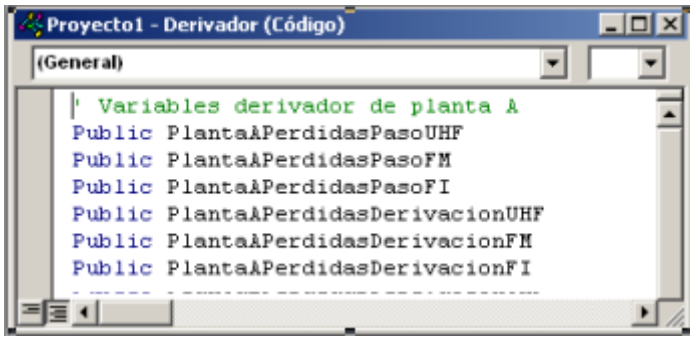

Figura 3.29. *Sección de código de declaración de variables*

La segunda de las finalidades del módulo consiste en la declaración de la función encargada de validar la información introducida sobre los derivadores en sus correspondientes formularios por los usuarios. Dicha función a la que se llamó ***ValidarInformación,*** se muestra parcialmente en la figura 3.30.

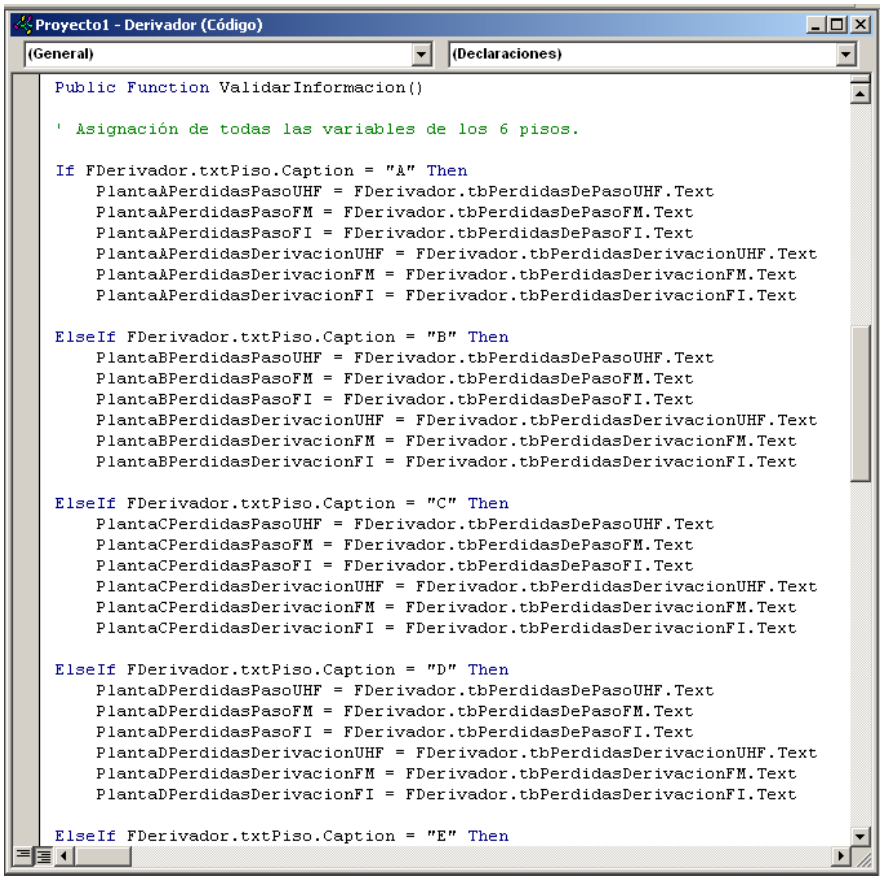

Figura 3.30. *Derivador.bas. Función ValidarInformacion.*

Y finalmente, la tercera de las misiones del módulo es la de imprimir los datos correspondientes sobre el formulario principal (*Fmain.frm*). Para ello se dispone de la función ***ImprimirInformacion*** cuyo código se muestra en la figura 3.31.

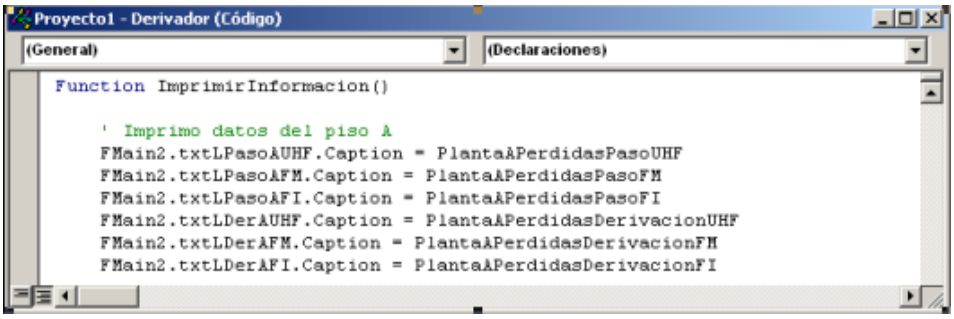

Figura 3.31. *Derivador.bas. Función ImprimirInformacion*

- ***"Distribuidor.bas":*** Análogamente al módulo derivador, el distribuidor tiene las mismas obligaciones: Declaración de variables, ValidarInformación e ImprimirInformación. Las figuras 3.32 – 3.34 muestran los códigos asociados a cada una de las citadas tareas.

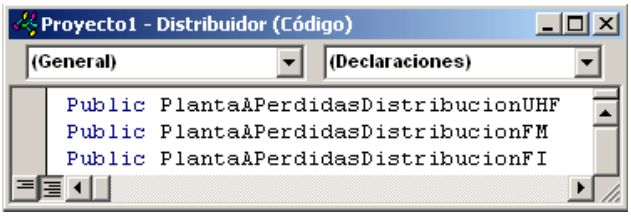

Figura 3.32. *Distribuidor.bas. Declaración de las variables de acceso público*

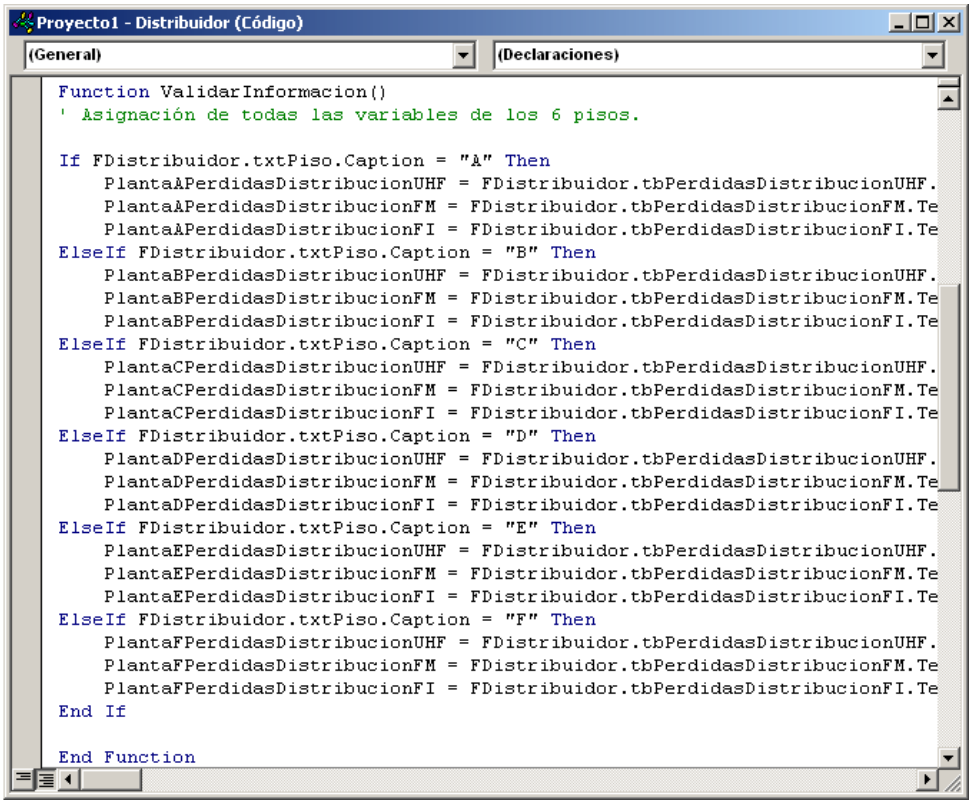

Figura 3.33. *Distribuidor.bas. Función ValidarInformación.*

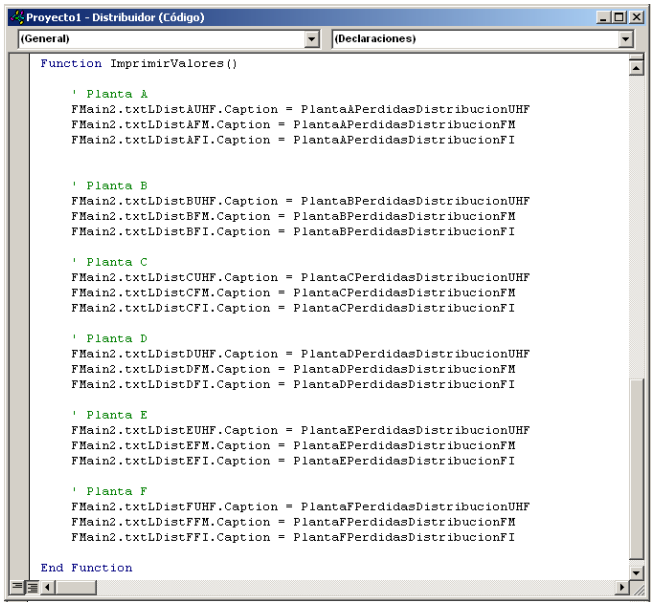

Figura 3.34. *Distribuidor.bas. Función ImprimirValores.*

- ***"PanelDeControl.bas":*** En este módulo la única función implementada a sido ***VerificarDatos*** cuya finalidad es, como su nombre indica, la de corroborar que todos los datos se encuentran dentro de los rangos permitidos por el programa. Su código se muestra en la figura 3.35.

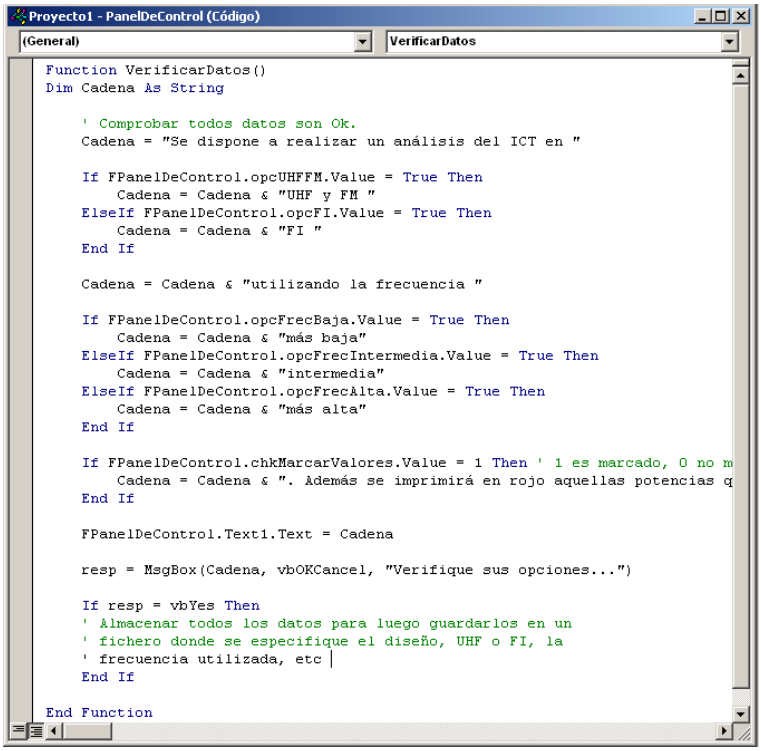

Figura 3.35. *PanelDeControl.bas. Función VerificarDatos.*

- ***"Proyecto.bas":*** Siguiendo con la tónica de trabajo anterior, el módulo "Proyecto.bas" tiene una parte encargada de la definición y declaración de las variables a utilizar y otra parte de verificación (figuras 3.36 y 3.37).

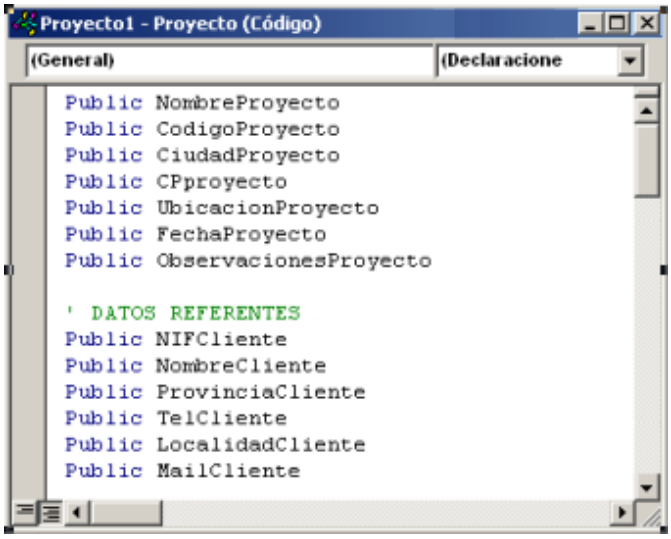

Figura 3.36. *Proyecto.bas. Declaración de variables de acceso público*

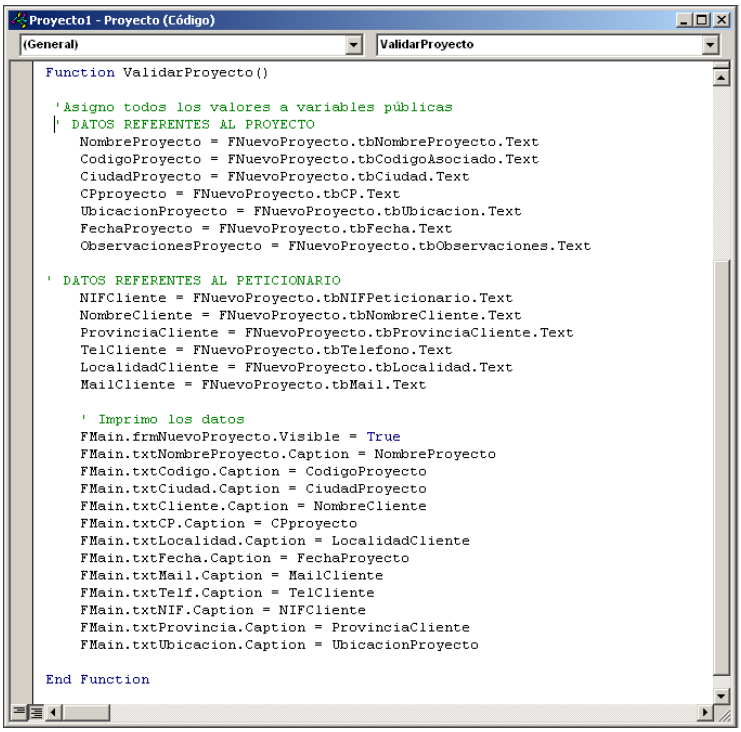

Figura 3.37. *Proyecto.bas. Función ValidarProyecto.*

- ***"RedDeDistribucion.bas":*** Nuevamente, aunque de modo parcial, la figura 3.38 muestra cómo el código del módulo se divide en la declaración de

variables y la verificación de los datos al igual que sucedía en los casos anteriores.

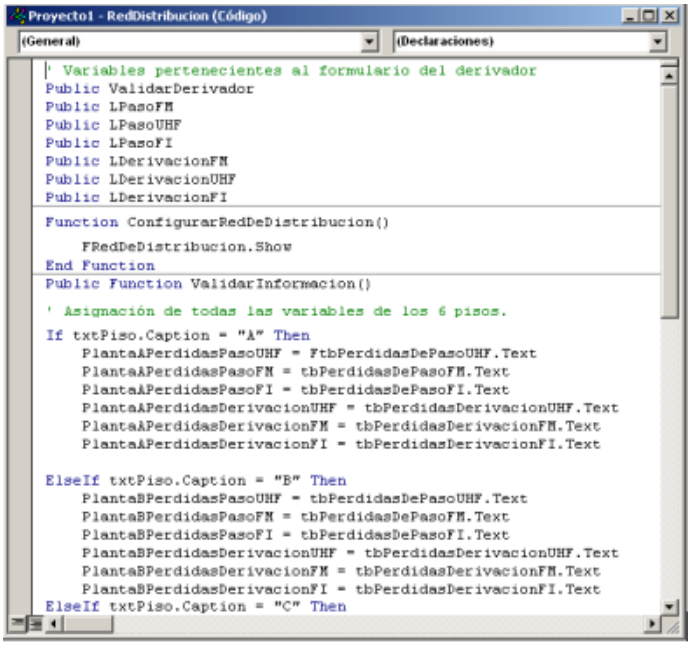

Figura 3.38. *RedDeDistribucion.bas. Función ValidarProyecto.*

- ***"SistemaDeCaptación.bas":*** Este módulo ya presenta una complejidad adicional a los anteriores, si bien se ha utilizado también una sección de declaración de variables (figura 3.39) y una única función encargada de verificar e imprimir datos en pantalla en tiempo de diseño como muestran las figuras 3.39 y 3.40.

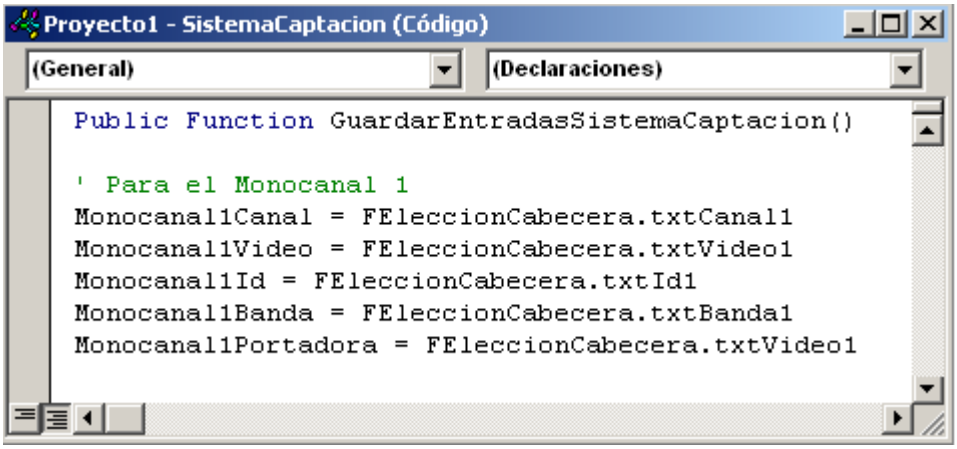

Figura 3.39. *SistemaDeCaptacion.bas. Bloque 1 de la Función .GuardarEntradasSistemaCaptacion*

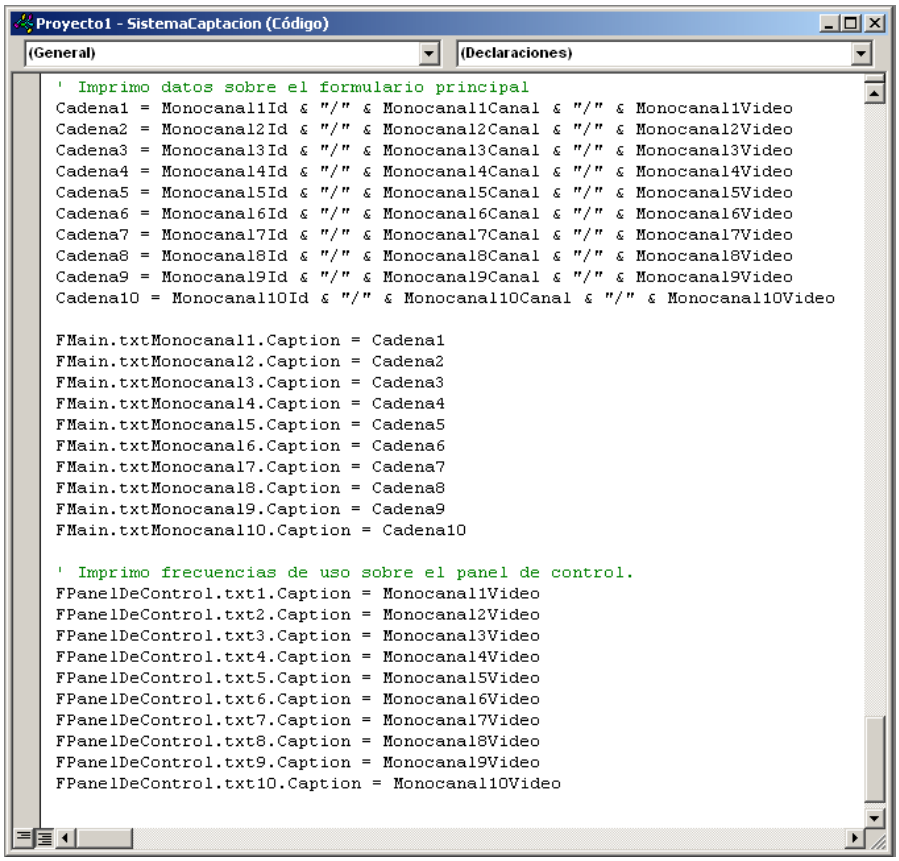

Figura 3.40. *SistemaDeCaptacion.bas*

- ***"Tomas.bas":*** Este módulo representa el último del tipo específico y consecuentemente se encuentra en primer lugar la declaración de variables de acceso público (figura 3.41).

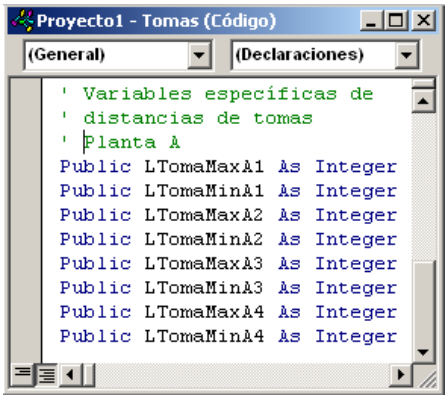

Figura 3.41. *Tomas.bas. Declaración de variables.*

Como en los casos anteriores se realiza tanto una verificación de la información introducida por el usuario (figura 3.41) como una impresión de estos sobre el formulario principal de la aplicación (figura 3.42 y 3.43).

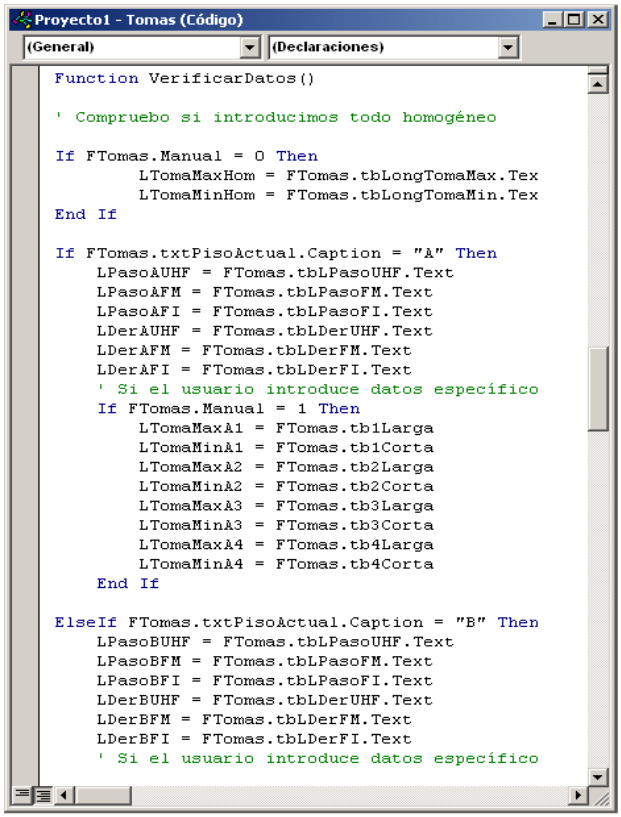

Figura 3.43. *Tomas.bas. Sección de la Función de Verificación de Información*

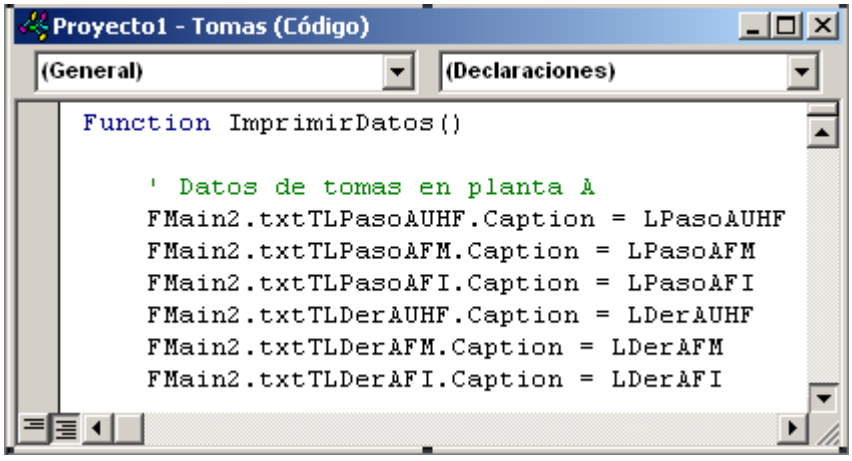

Figura 3.43. *Tomas.bas. Sección de la Función de Imprimir Información*

Con esto finaliza el estudio de los módulos sencillos, donde como se ha visto que la misión de éstos es, la declaración de las variables de uso, la verificación de la información introducida por el usuario en los correspondientes formularios de la aplicación y la impresión de información donde se solicite por el programa.

3.3.2.2. Módulos generales.

A continuación se estudiarán los módulos restantes "*funciones.bas*" y "*datos.bas*", que resultan mucho más complejos. En primer lugar abarcaremos el módulo "*datos.bas*" por ser el menos complejo de los dos y aunque más complejo que los anteriores, también dispone de una sección de declaración de variables, como puede verse en la figura 3.44.

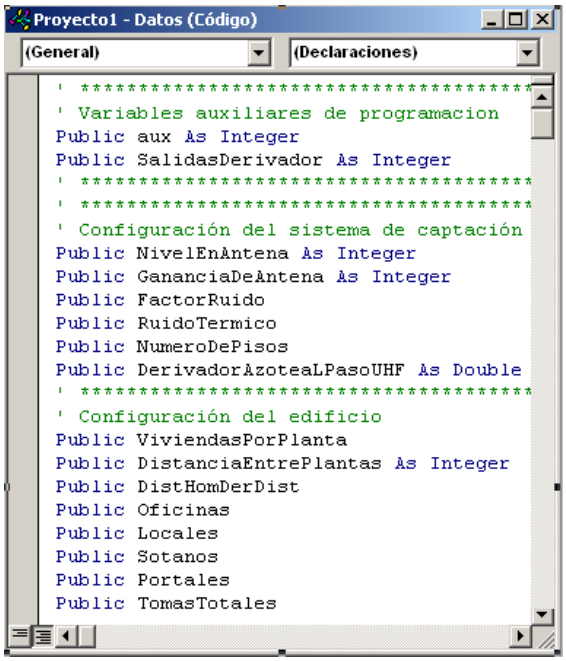

Figura 3.44. *Datos.bas. Declaración de variables*

La primera de las funciones que nos encontramos en el módulo bajo estudio se denomina ***DibujaPlantasEdificio*** (figura 3.45) y su finalidad es la de imprimir sobre el esquemático el número de alturas seleccionado por el usuario (número que

puede estar comprendido entre 1 y 6 plantas) y el número de viviendas por planta cuyas posibilidades abarcan desde 1 a 4 viviendas.

Es importante remarcar que tanto la selección del número de alturas como el número de viviendas en cada una de ellas está limitado en esta primera versión de *Calcicat*, estando prevista su ampliación en futuras versiones.

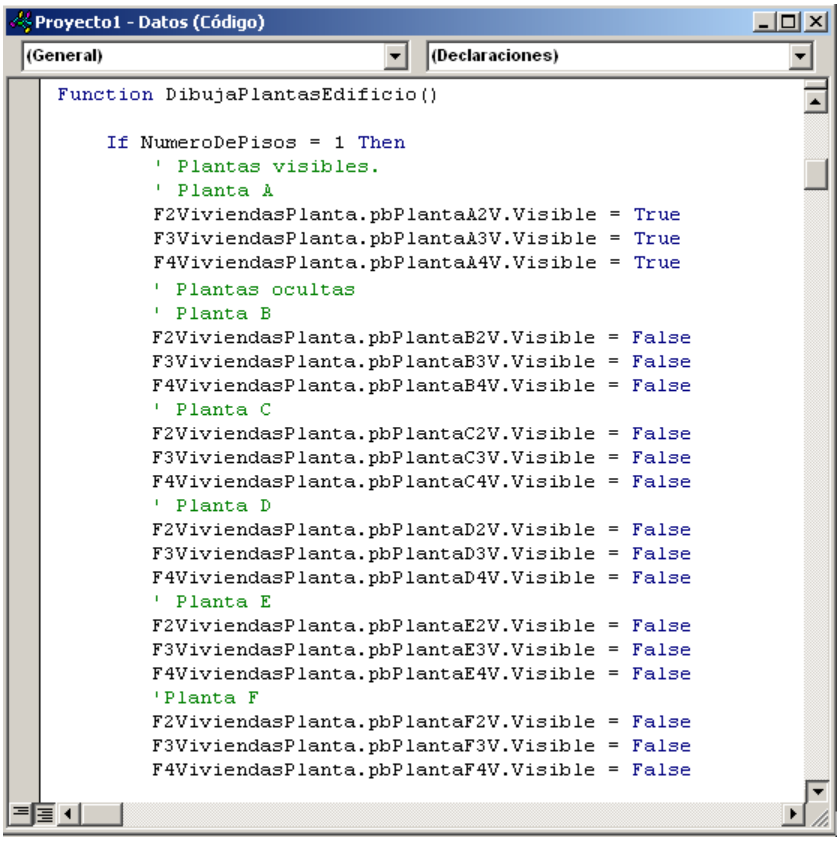

Figura 3.45. *Datos.bas. Sección de la función encargada de dibujar las plantas en el esquemático*

La función mostrada basa su funcionamiento en la ocultación de imágenes y la superposición de otras de tal modo que al final de la impresión se construye una edificación totalmente particularizada. De este modo, el resultado visible en pantalla, al que se le denomina esquemático, no es más que una mera superposición de imágenes y ocultación de otras estructuras funcionales básicas como, por ejemplo, viviendas, cargas de adaptación, cableado de troncal de bajada, etc. El resultado final es la generación de cualquier tipo de inmueble comprendido entre 1 altura con 2

viviendas por planta (figura 3.46) hasta 6 alturas con 4 viviendas por planta (figura 3.47).

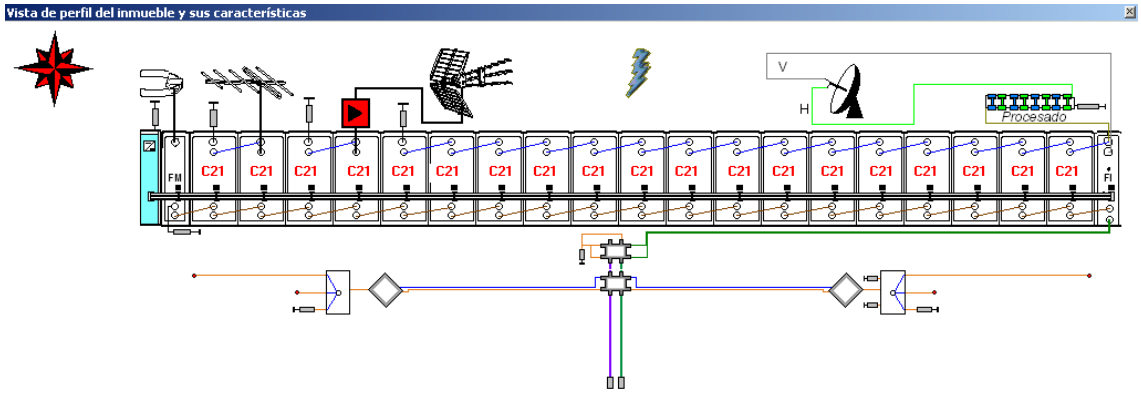

Figura 3.46. *Resultado de la ejecución de la función DibujaPlantasEdificio para 2 viviendas por planta y 1 altura.*

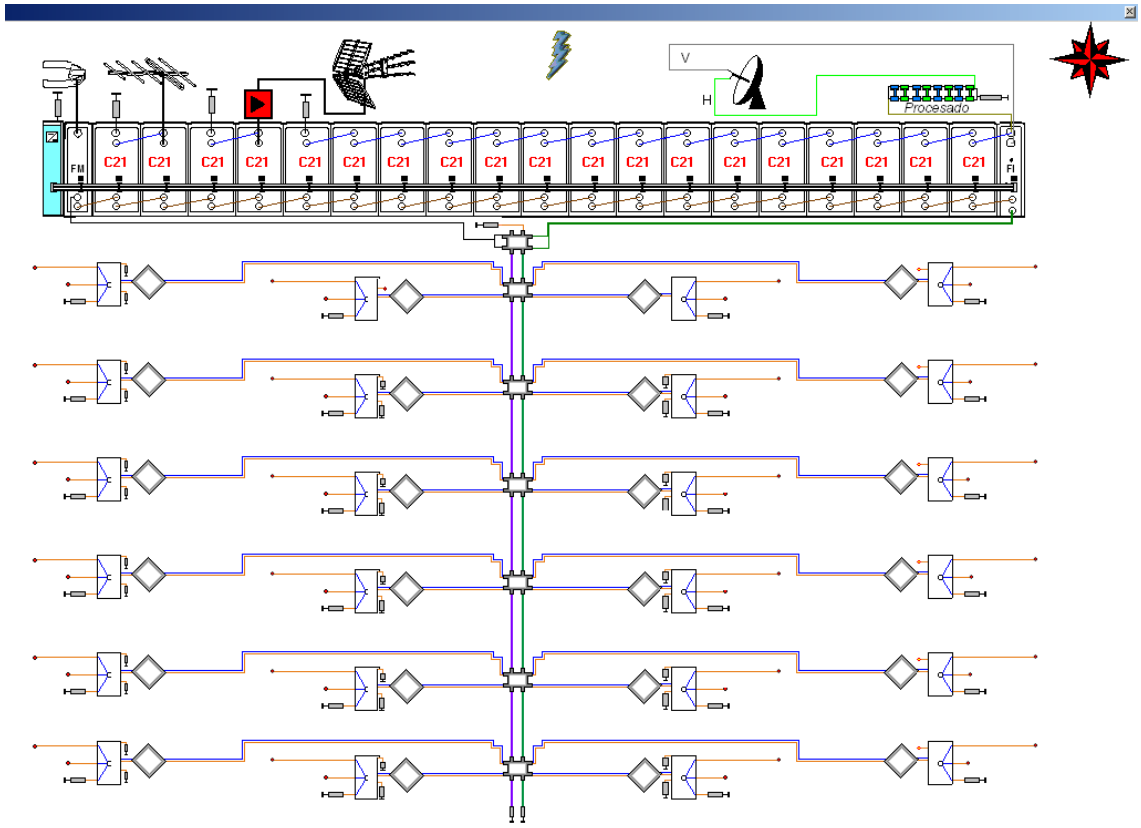

Figura 3.47. *Resultado de la ejecución de la función DibujaPlantasEdificio para 4 viviendas por planta y 6 alturas.*

La siguiente función del módulo es ***VerificarCaptacion,*** cuya misión es validar información sobre el sistema de captación, como los niveles a la salida de las antenas.

Otra función del módulo de datos es la de ***VerificarEdificio,*** cuya finalidad es la de verificar que el edificio que *Calcicat* se dispone a generar cumple todas las exigencias internas que el programa exige. Además, dicha función también se encarga de salvaguardar toda la información pertinente a cálculos específicos, restando homogeneidad al diseño del ICT típico. Esto constituye una de las grandes ventajas de *Calcicat* frente al resto de programas de cálculos de ICT. La figura 3.48 muestra una sección de código de la función citada.

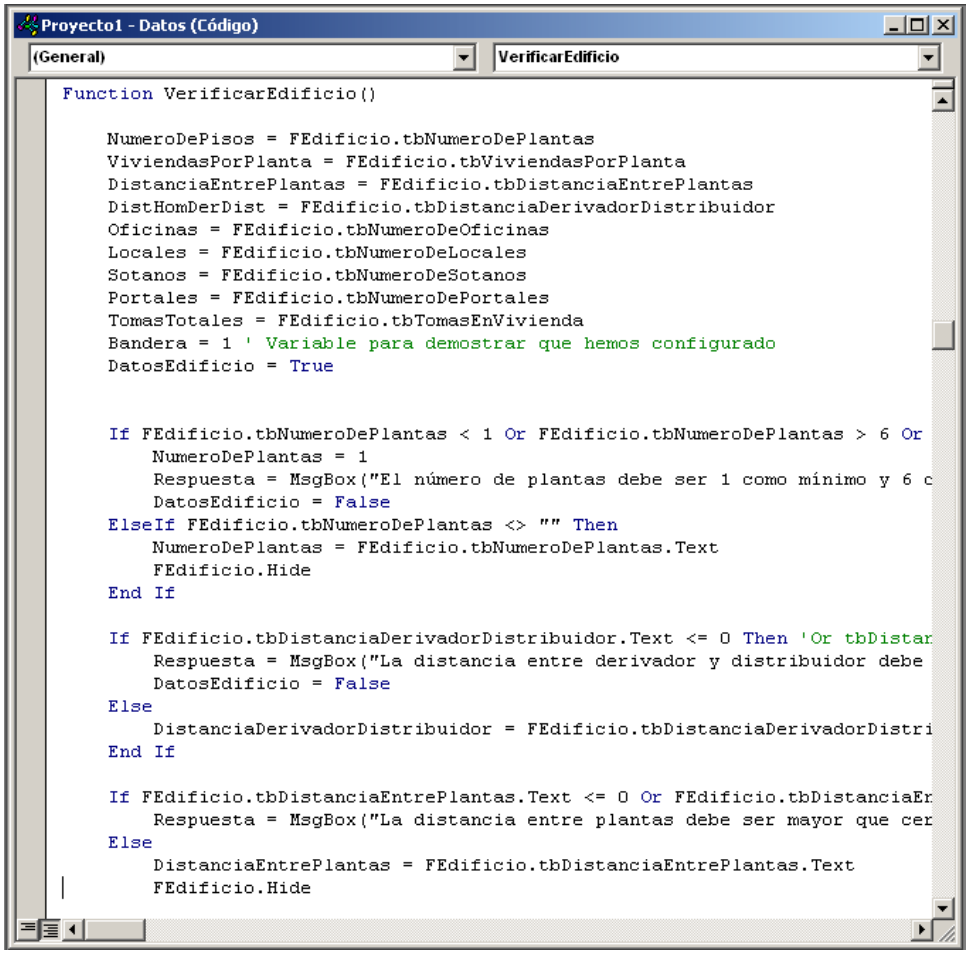

Figura 3.48. *Sección de código de la función VerificarEdificio*

Quizás la parte del módulo más importante es la función ***CalcularICT;*** pues de ella se deriva la obtención de todos los niveles de señal en derivadores, paus, tomas de usuario, etc.

El modo en que dicha función ha sido implementada es crucial para la interacción con el resto del programa (tanto en módulos como en formularios) ya que en múltiples ocasiones se diseñará el ICT para distintas frecuencias y, por lo tanto, ejecutaremos la citada función un determinado número de veces con distintos valores de frecuencia de trabajo. Lo mismo sucederá a la hora de cambiar cualquier tipo de dispositivo que comprenda el ICT, como un derivador de planta, un distribuidor, las tomas, longitudes en las viviendas e incluso el nivel de señal del que se dispone en la azotea.

Es por todo ello que dicha función es, casi sin lugar a dudas, la más importante de este módulo. La forma en la que ha sido implementada ha seguido los siguientes requisitos:

- Ser óptima en tiempo de diseño.
- Minimizar el número de objetos residentes en pantalla.
- Clarificar operaciones para posibles ampliaciones.

Con estos tres requisitos, a la hora de introducir detalles como un derivador tras el sistema de cabecera con sus pérdidas de paso y derivación, el sistema recalcula todos los niveles de señal fácilmente.

El modo de cálculo se ha realizado a partir de unas constantes interdependientes, es decir, la constante N (Cte(N)) depende de la Cte(N-1) y así sucesivamente hasta llegar al comienzo de los cálculos donde no existen dependencias.

La figura 3.49 muestra sólo el comienzo de unos pocos cálculos que se corresponden con la señal en antena, las pérdidas tras los monocanales y al comienzo de la troncal de bajada.

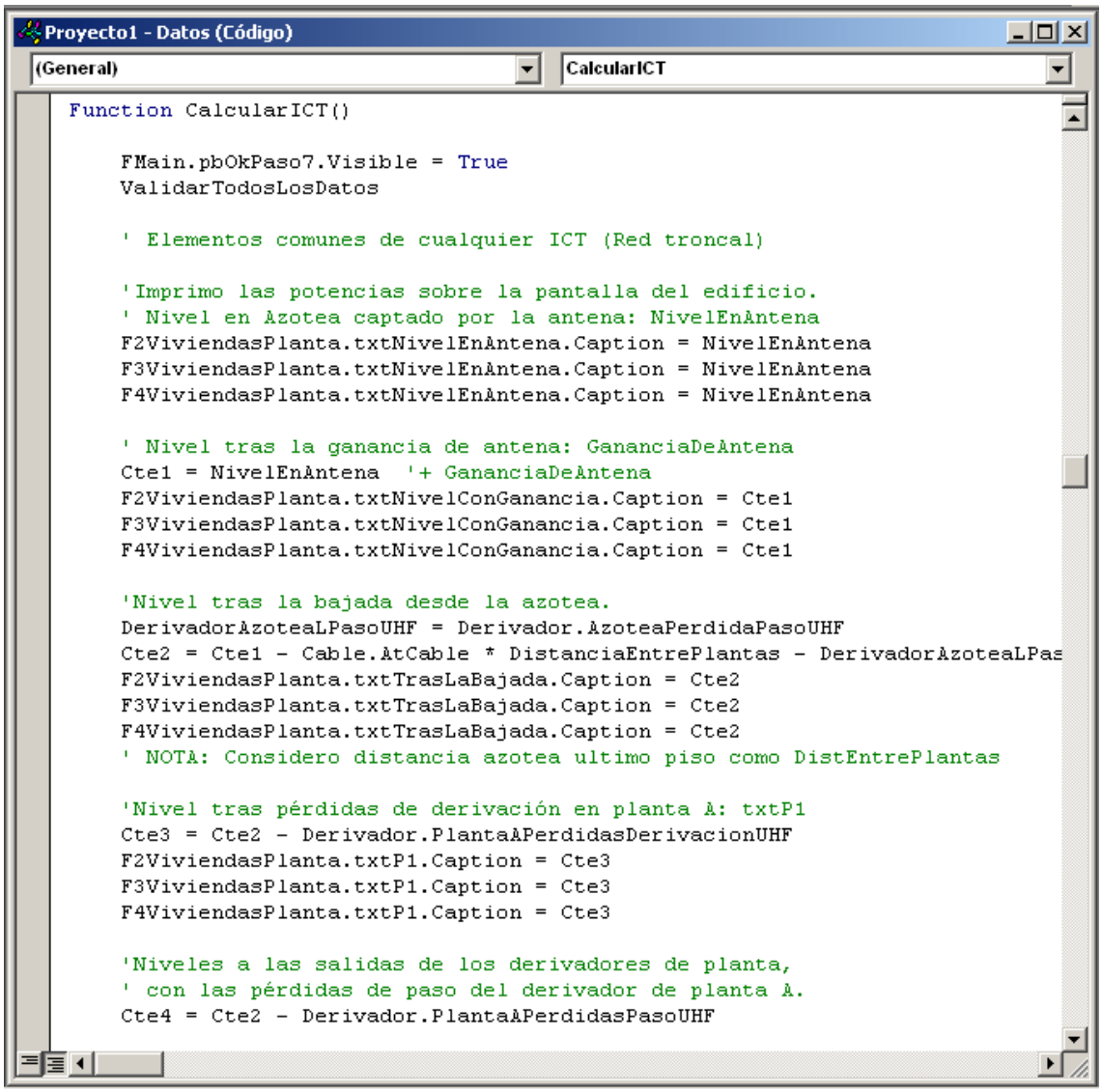

Figura 3.49. *Sección de código de la función CalcularICT*

El lector debe quedarse con la idea de que para realizar cálculos a distintas frecuencias, con distintos elementos, con variaciones de elementos del inmueble, etc no serán necesarias funciones adicionales ya que ***CalcularICT*** ha sido diseñada de un modo eficiente y flexible que permite realizar todas estas posibilidades.

Para finalizar el módulo, existen dos funciones más (***ValidarTodosLosDatos*** y ***GuardarDatosMonocanales***) mostradas en la figura 3.50 que no son más que funciones auxiliares cuando se trabaja en múltiples puntos de frecuencia.

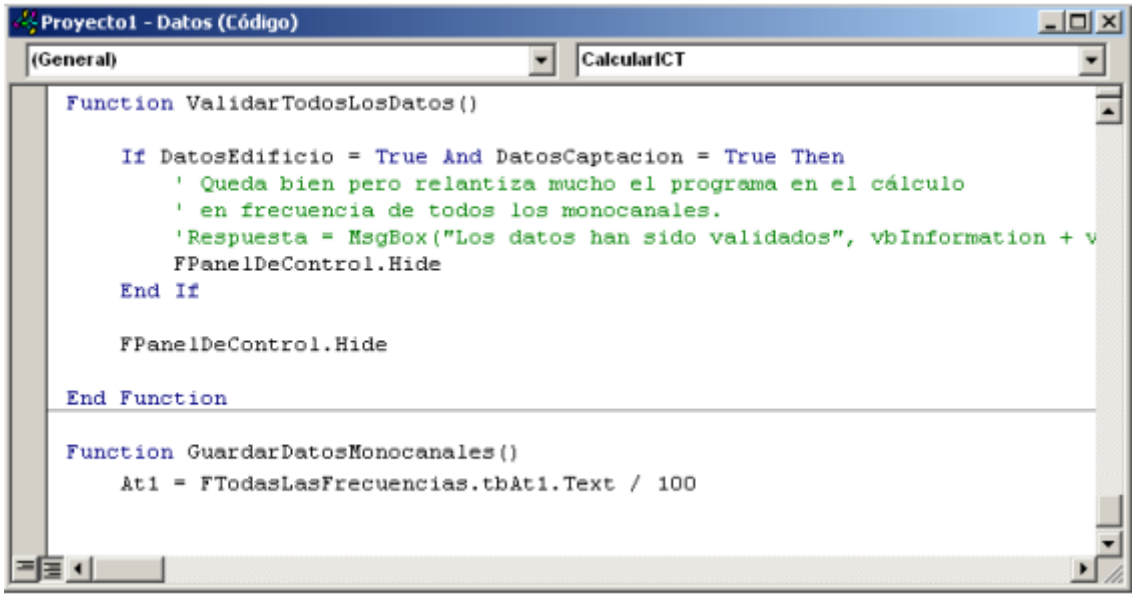

Figura 3.50. *Funciones finales del módulo de datos.*

Como puede verse, la función ***GuardarDatosMonocanales*** convierte los datos introducidos por el usuario en dB/100m a dB/m, ya que suele ser la primera la usada por fabricantes de dispositivos de Infraestructuras Comunes de Telecomunicaciones.

El siguiente de los módulos importantes es "***Funciones.bas***", donde se declaran e implementan la mayor parte de las funciones utilizadas por el programa. La figura 3.51 muestra los comienzos de este módulo junto con la inicialización de variables y de su primera función ***ImprimirMonocanales***.

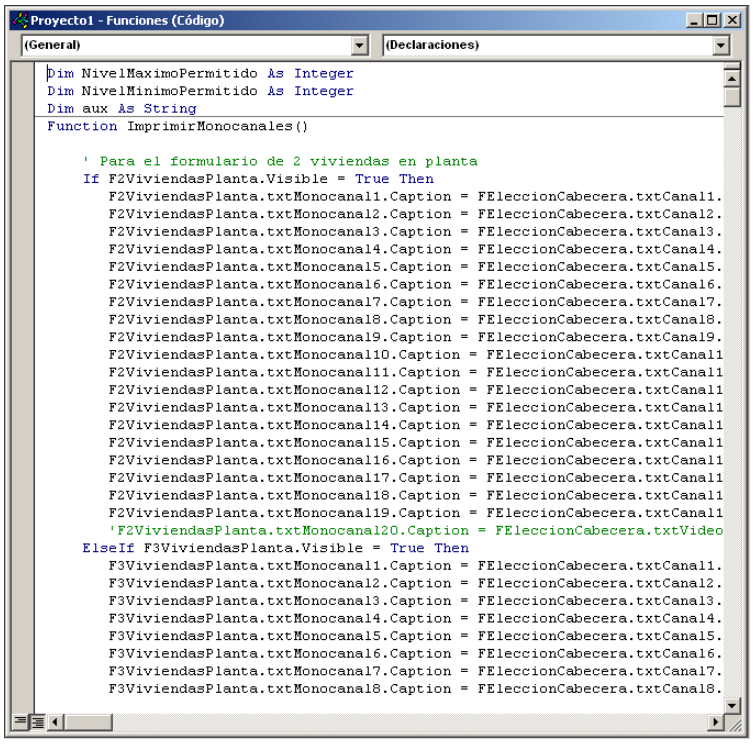

Figura 3.51. *Módulo funciones.bas*

La citada función tiene como única finalidad supervisar que todas las impresiones en pantalla de los monocanales, incluyendo formulario principal y esquemático, sean las adecuadas. En caso contrario, la propia función es autosuficiente para regenerar las citadas impresiones obteniendo, en todo momento, un nivel de mando superior a las funciones locales que puedan existir.

Seguidamente en el módulo se encuentra la función ***BalanceDeSenal,*** cuya misión es realizar un balance de las señales obtenidas en el esquemático. De este modo, si existen niveles que no cumplen los requisitos de la reglamentación serán marcados en rojo para que el diseñador pueda identificarlos fácilmente. La figura 3.52 y 3.53 muestran una sección de código asociado y el efecto respectivamente.

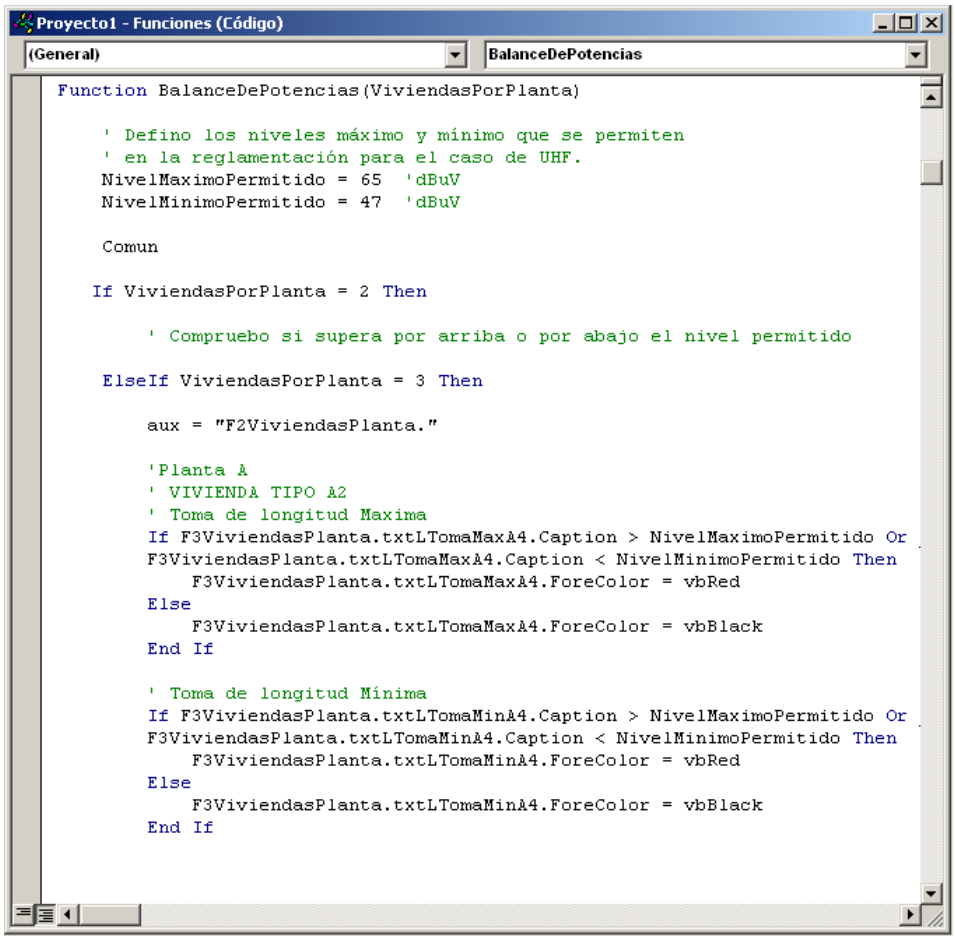

Figura 3.52. *Módulo funciones.bas. Sección de código de la función Balance de potencias.*

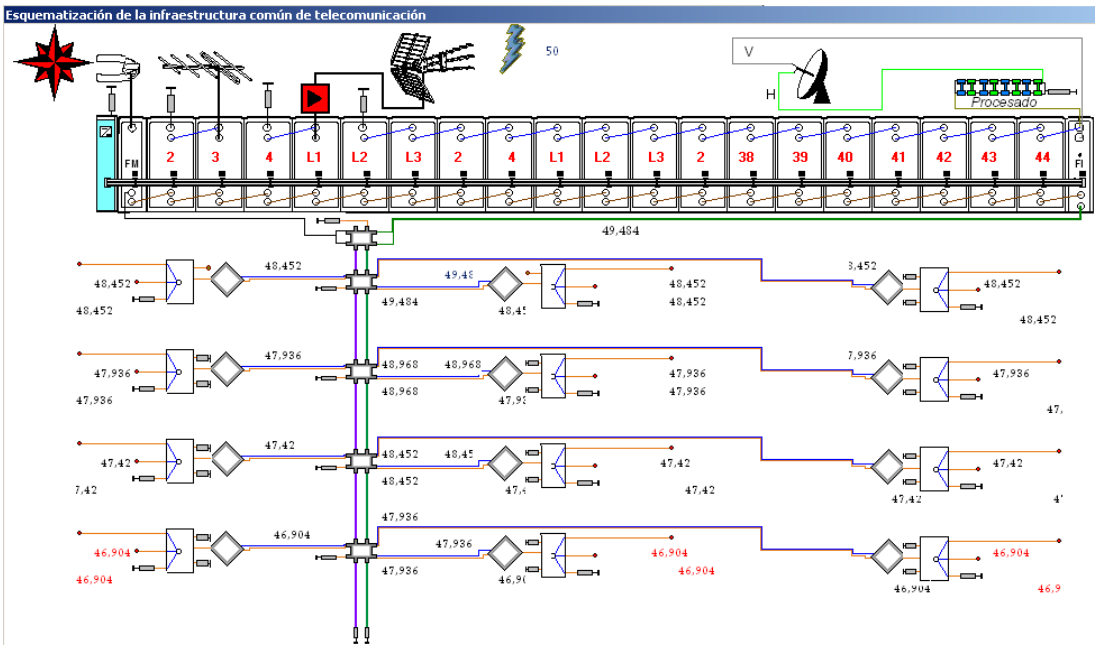

Figura 3.53. *Efecto de la función de balance de potencias.*

La siguiente función del módulo trata aspectos de carácter común de cálculos, tanto para el balance de potencias como para las funciones encargadas de dibujar el esquemático, cálculos de gráficas que equilibran la potencia, etc. Dicha función se denomina ***Comun*** y una parte de su código se muestra en la figura 3.54:

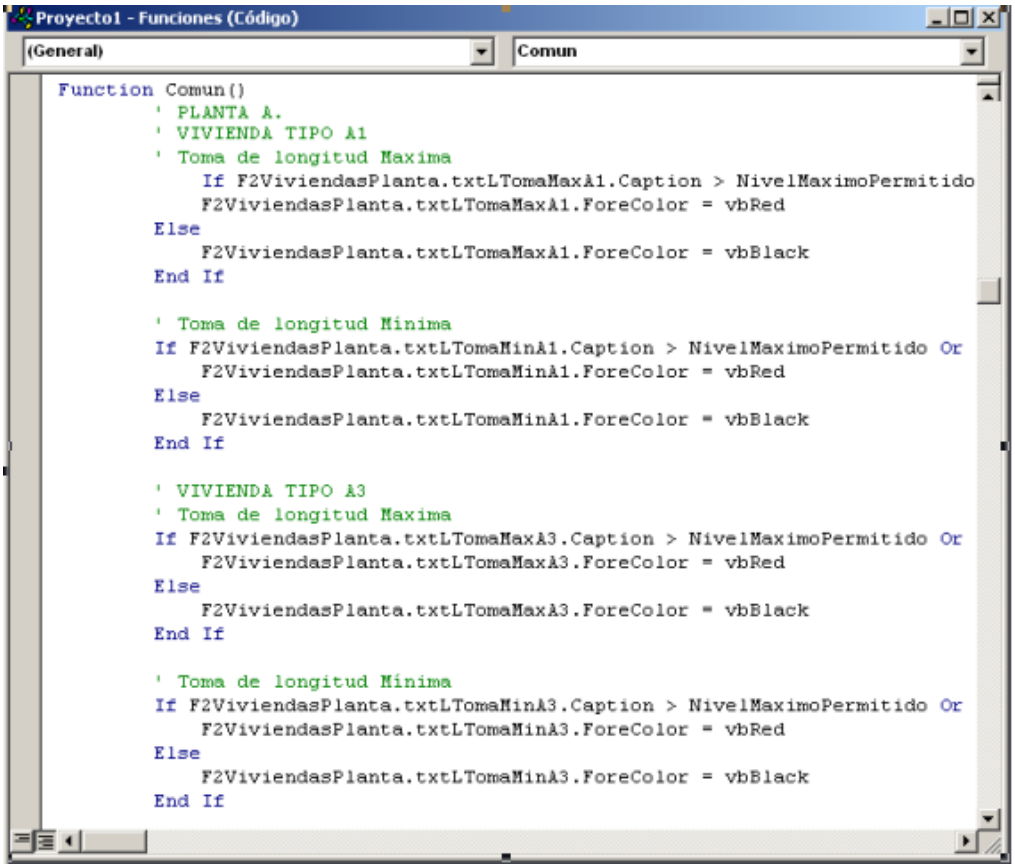

Figura 3.54. *Función común del módulo de funciones*

Esta función, lejos de parecer insignificante, es la más grande dentro del módulo, ocupando más de un 60% del espacio total del módulo "***funciones.bas***". Si se desea conocer el código en profundidad no hay más que recurrir al código adjunto al proyecto o al interior del propio programa *Calcicat*, donde se podrá encontrar el código íntegro de cada formulario, función y módulo anteriormente descritos.

Por otro lado, y como puede haberse comprobado hasta el momento, el programa ha sido comentado de tal modo que con un estudio sencillo el lector es capaz de determinar la finalidad y funcionalidad del código leído.

3.3.3 Programación de eventos.

Resta comentar algunos detalles de aspecto general de la programación de eventos asociada a cada uno de los formularios presentados en los puntos anteriores.

En principio, los formularios F2ViviendasPlanta, F3ViviendasPlanta y F4ViviendasPlanta poseen la misma estructura, siendo sus funciones asociadas a eventos las siguientes:

Función	Objetivo
• *Borratodo*	Adecúa el interfaz gráfico en función de la elección escogida: Modo Asistente o Experto
• *DibujaXViviendasPlanta*	Dibuja las viviendas correspondientes en cada planta (depende del .frm).
• *VerOcultar*	Modifica objetos para dar sensación de realismo de diseño en tiempo real.
• *Calcular*	Calcula los niveles de señal en toma corta y toma larga para cada una de las viviendas del inmueble. Esta función está supervisada por la función ***Calcula*** presentada anteriormente en el módulo de datos.
Subrutinas	**Objetivo**
• *SalidasDerivador*	Independiza las salidas del repartidor propio de cada vivienda (ver figura 3.56)
• *DerivadorSuperior*	Determina/Establece las características del derivador troncal de bajada.
• *Recalcular*	Encargada de actualizar todas las variables de modo pertinente y al aplicar la función de CalcularICT presentada los nuevos datos estén afectados por los nuevos cambios. (figura 3.57)
• *Form_Load*	Adecua el interfaz gráfico, carga los controladores, comprueba el sistema, etc. Es especialmente importante en *el Fmain.frm*.

La figura 3.55 muestra código referenciado en la subrutina *pbSalidasDerivador_Clic().*

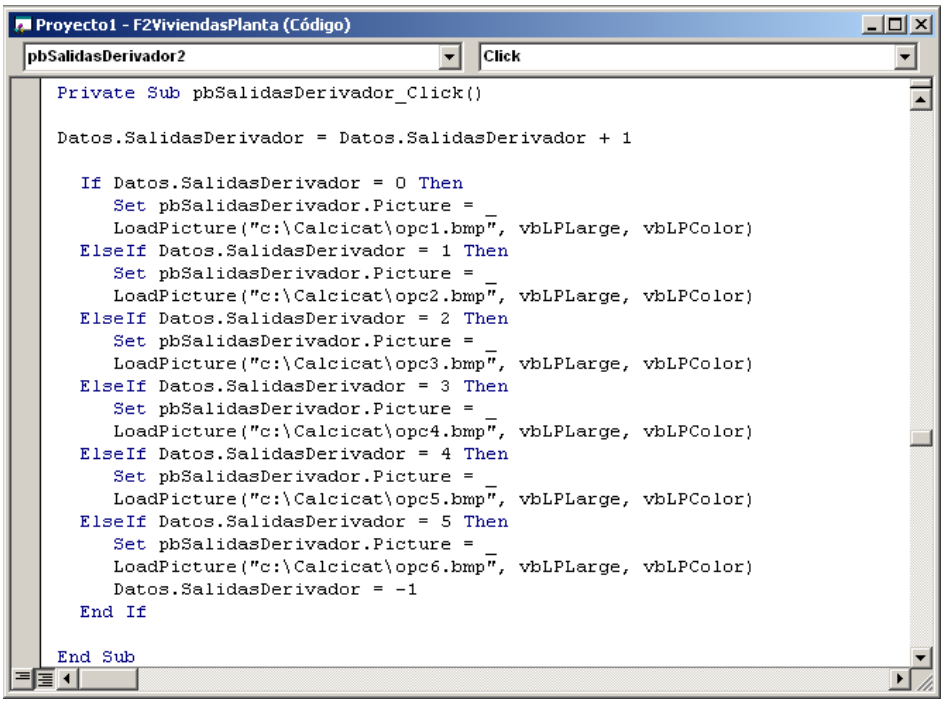

Figura 3.55. *Evento asociado a la subrutina salidas del derivador*

El evento se particulariza para cada tipo de repartidor o derivador, según sea el caso, que aparece en el esquemático que aunque a simple vista pueden parecer sólo 4, en realidad llegan a ser 4*6 = 24, siendo 6 el número de todas las posibilidades que pueden adquirirse en cada caso.

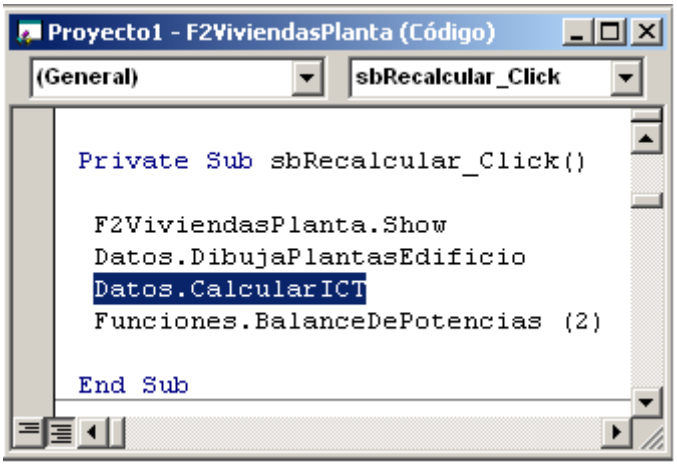

Figura 3.56. *Código asociado al evento de recalcular.*

Puede verse como la subrutina ***sbRecalcular_Clic()*** depende de la función ***CalcularICT*** del módulo "*Datos.bas*" citada anteriormente.

Pasando a estudiar el código de eventos asociado a otros formularios aparece "*FayudaTotal.frm*" y "*FacercaDe.frm*". La programación de ésta última se muestra a en la figura 3.57:

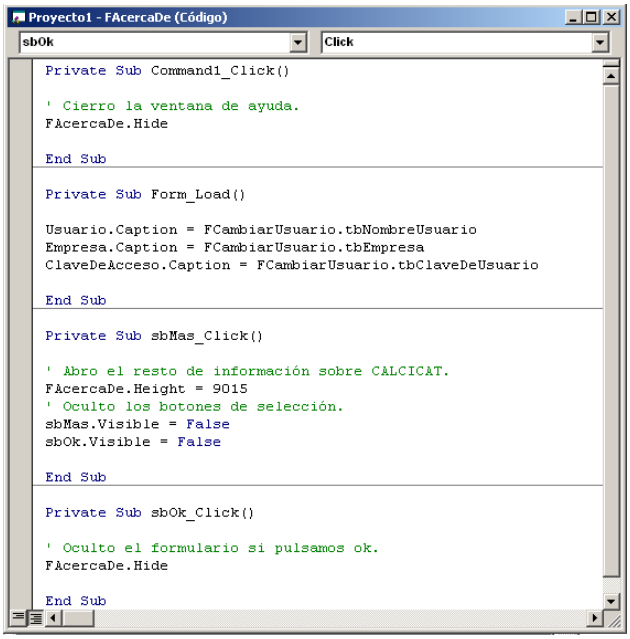

Figura 3.57. *Programación del formulario FacercaDe.frm*

En "*FAtenuaciones.frm*" la subrutina más importante es ***sbOk_Clic()*** cuyo bucle se muestra en 3.58 donde se calcula nuevamente el esquemático con la función ***CalculaICT()*** del módulo de "*Datos.bas*".

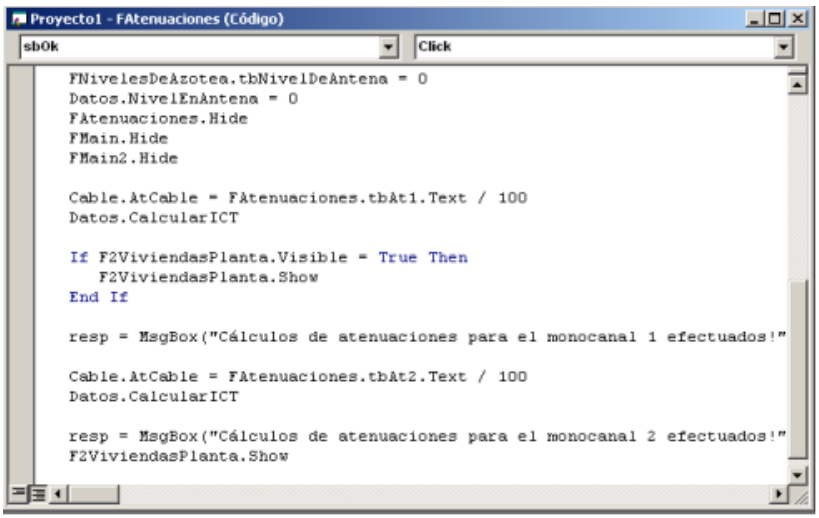

Figura 3.58. *Bloque más relevante del formulario FAtenuaciones.frm*

Como puede verse en esta sección de código, éste no sólo depende de ***Datos.CalcularICT(),*** sino que también se alteran las variables *Cable.AtCable* y *Datos.NivelEnAntena.*

Los formularios "*FcambiarDeUsuario.frm*" y "*FconfigurarCable.frm*" carecen de interés en lo referente al cálculo del esquemático y, por lo tanto, se obvia detallar su código. El formulario "*FConfigurarCableManual.frm*" muestra parte de su código en la figura 3.59:

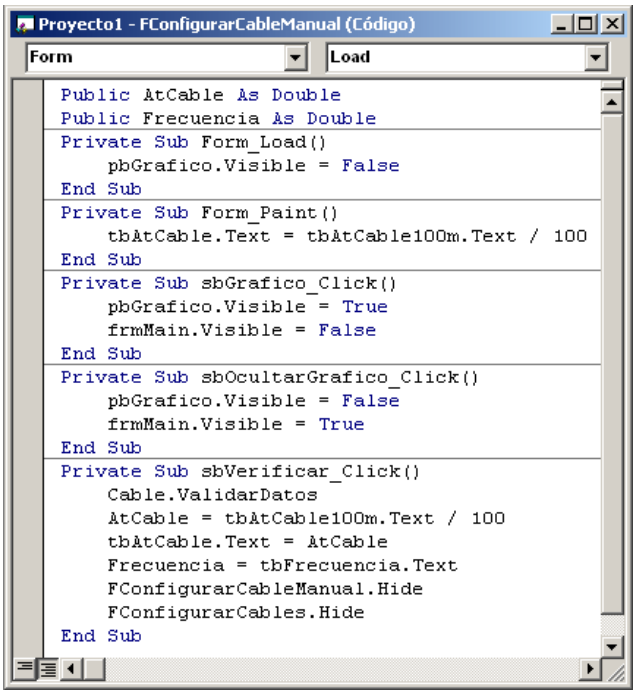

Figura 3.59. *Código del formulario FConfigurarCableManual.frm*

Su finalidad reside en verificar, adecuar gráficos bien sea mostrándolos por pantalla u ocultándolos, cargar adecuadamente el formulario y actualizarlo debidamente ante cambios introducidos por el usuario.

Siguiendo el recorrido de formularios, "*FDerivador.frm*" posee una parte importante de declaración de variables seguida de una función de validar información controlada por el módulo ***Derivador.ValidarInformacion()*** y una

sección de eventos (figura 3.60) asociados al control de la base de datos "*Derivadores.mdb*".

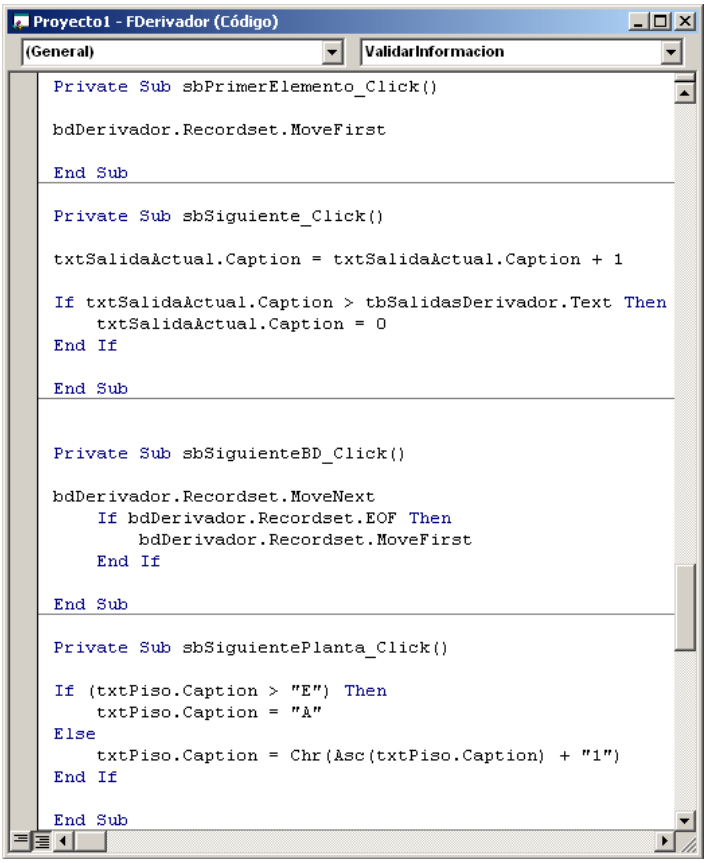

Figura 3.60. *Código asociado a eventos sobre la base de datos "Derivadores.mdb"*

El formulario *"FDistribuidor.frm"* presenta un funcionamiento similar al anterior. En la figura 3.60 se ha destacado, por completar la información anterior, una sección de código de almacenamiento en memoria muy interesante que aporta *Calcicat*.

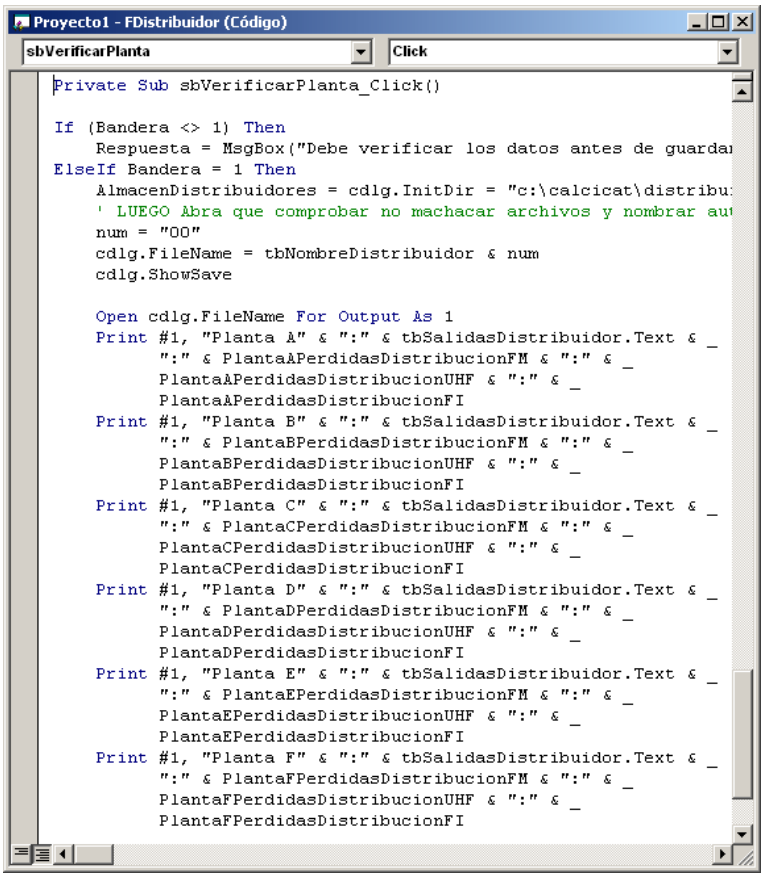

Figura 3.61. *Código asociado de almacenamiento en memoria en el formulario "FDistribuidor,frm"*

El siguiente formulario ("Fedificio.frm") trata sobre la configuración del edificio. Su código es bastante extenso ya que dispone de las siguientes partes:

- Declaración de variables
- **Subrutina *Form_Load***: evento asociado a la carga de un formulario.
- **Subrutina *ManualDerivadorDistribuidor***: almacena variables tanto de modo homogéneo como en casos particularizados, comprobando en todo momento especificaciones y restricciones software; la figura 3.62 muestra parte de este código.
- ***ValidarDatos*** y otras funciones similares ya conocidas por el lector.

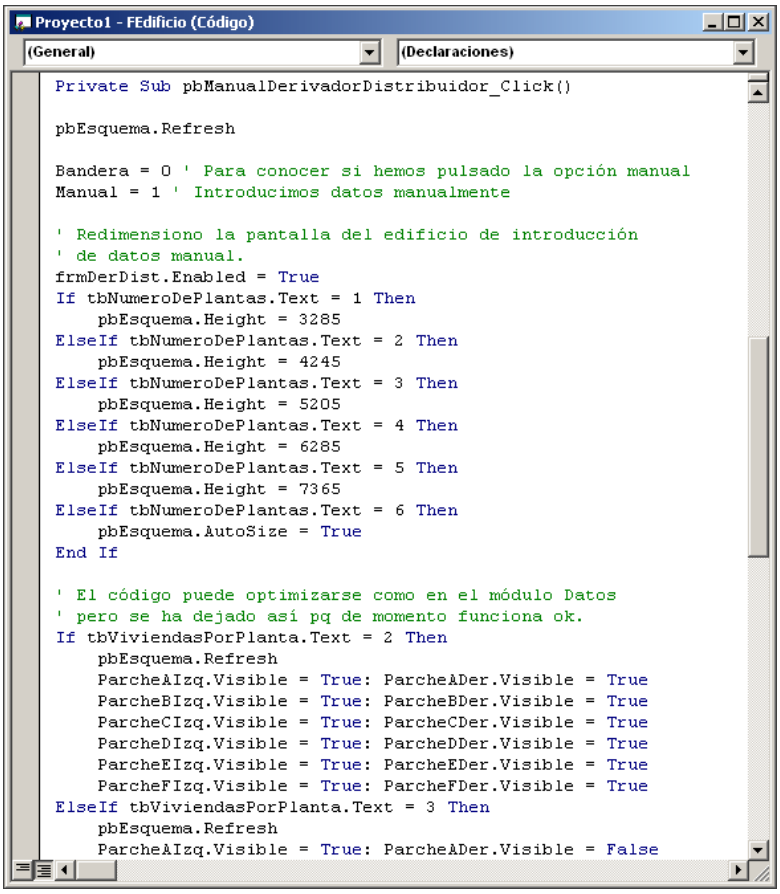

Figura 3.62. *Código asociado a SubrutinaManualDerivadorDistribuidor*

"*FeleccionCabecera.frm*" es uno de los módulos más complejos de *Calcicat*, no sólo por sus posibilidades sino también por su interactividad con el usuario. Constituye, en cuanto a tamaño de código más del 35% del total de código de eventos siendo sus partes más importantes las citadas a continuación:

- Función ***CargarDatos()***: como el nombre indica su finalidad es cargar datos previamente almacenados sobre el formulario asociado; a dicha tarea se adjunta el código mostrado en las figuras 3.63 y 3.64.

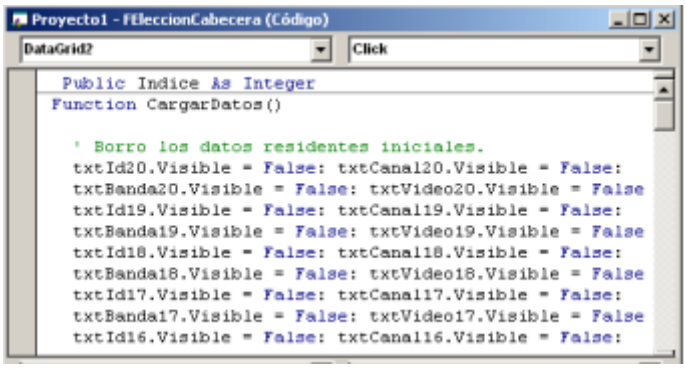

Figura 3.63. *Código asociado a cargar datos (Parte 1)*

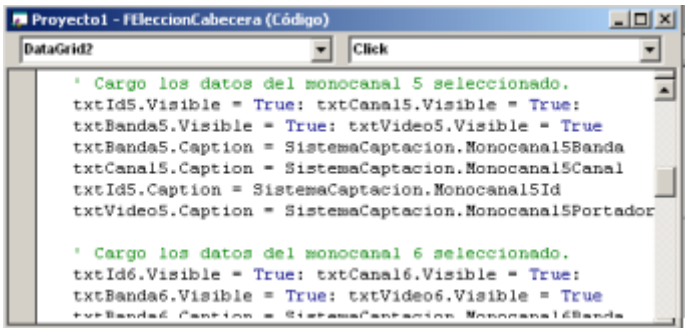

Figura 3.64. *Código asociado a cargar datos (Parte 2)*

- Función ***Reset()***: encargada de inicializar el sistema de selección de monocanales en el momento en que lo desee el usuario (figura 3.65).

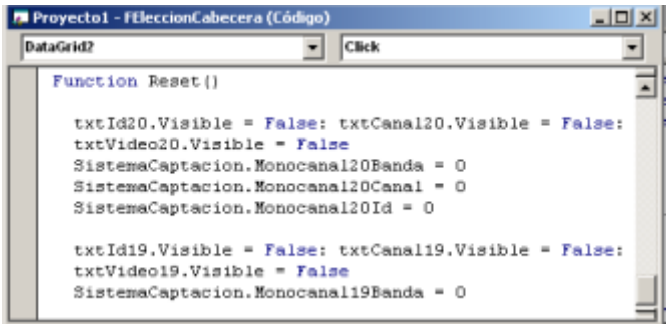

Figura 3.65. *Función Reset*

Otro bloque de código interesante es el encargado de controlar las pestañas de selección donde se muestran los elementos de la base de datos *"Canales.mdb"* y donde se pueden realizar consultas SQL con la finalidad de determinar aquellas características que cumplen nuestras exigencias. Dicho control se realiza a partir de la sección de código mostrada en la figura 3.66.

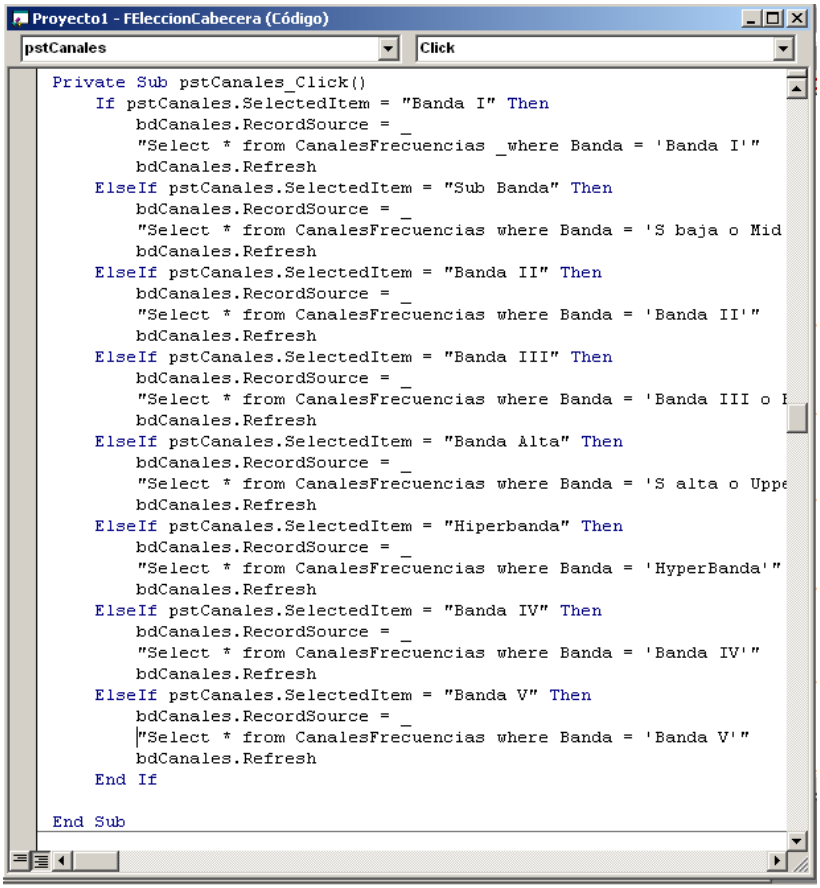

Figura 3.66. *Control sobre la base de datos mediante el objeto pstNew01*

La típica verificación de datos está controlada, como sucedía en los casos anteriores, por el módulo "*Datos.bas*" y cuya misión es idéntica a sus predecesoras, siendo su código asociado el de la figura 3.67.

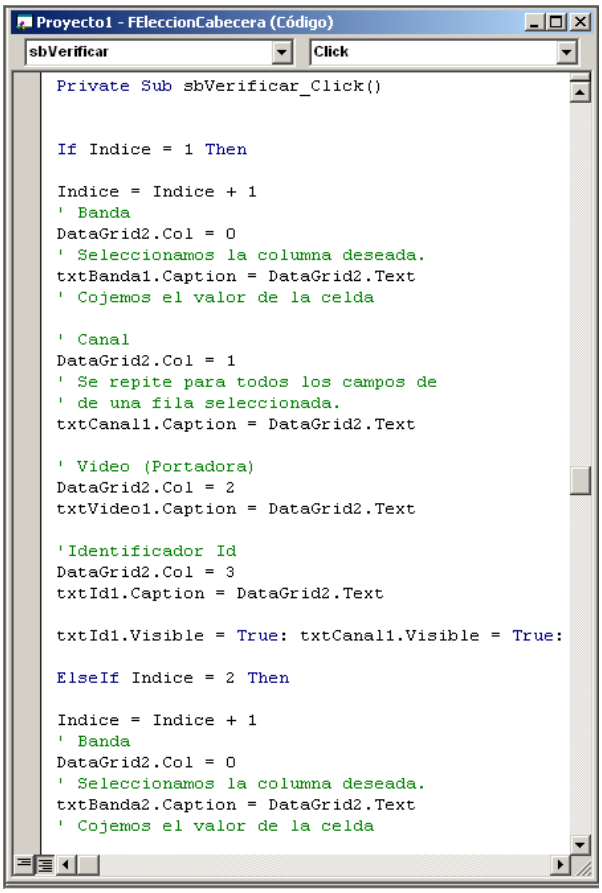

Figura 3.67. *Verificación de datos de monocanales*

El formulario encargado de representar gráficas que ayuden al diseñador a cuantificar el equilibrio en el reparto de señal entre las distintas tomas del inmueble se denomina *"Gráficas.frm"*. A dicho formulario se le asocia también uno de los códigos más extensos de la parte de eventos representando, cerca del 25% del total. Sus partes fundamentales son:

- Inicialización de variables.
- Función ***AdaptarAlNumeroDePlantas***.
- Función ***Dibuja():*** encargada de realizar gráficas en función de los datos adquiridos.
- Función ***Dibuja2():*** complementa a la función anterior ampliando las posibilidades de representación de manera considerable.

En las figuras 3.68 a 3.73 se puede observar parte del código a este formulario.

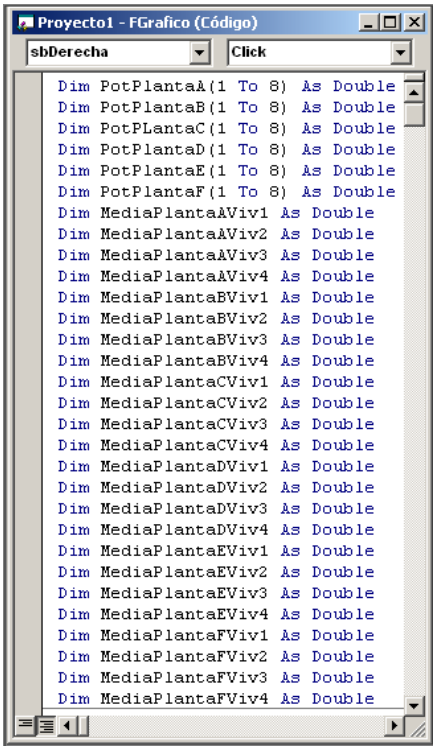

Figura 3.68. *Inicialización de variables.*

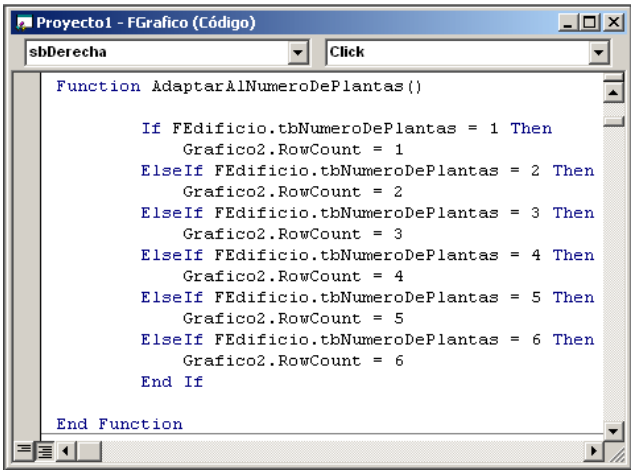

Figura 3.69. *Función AdaptarAlNumeroDePlantas*

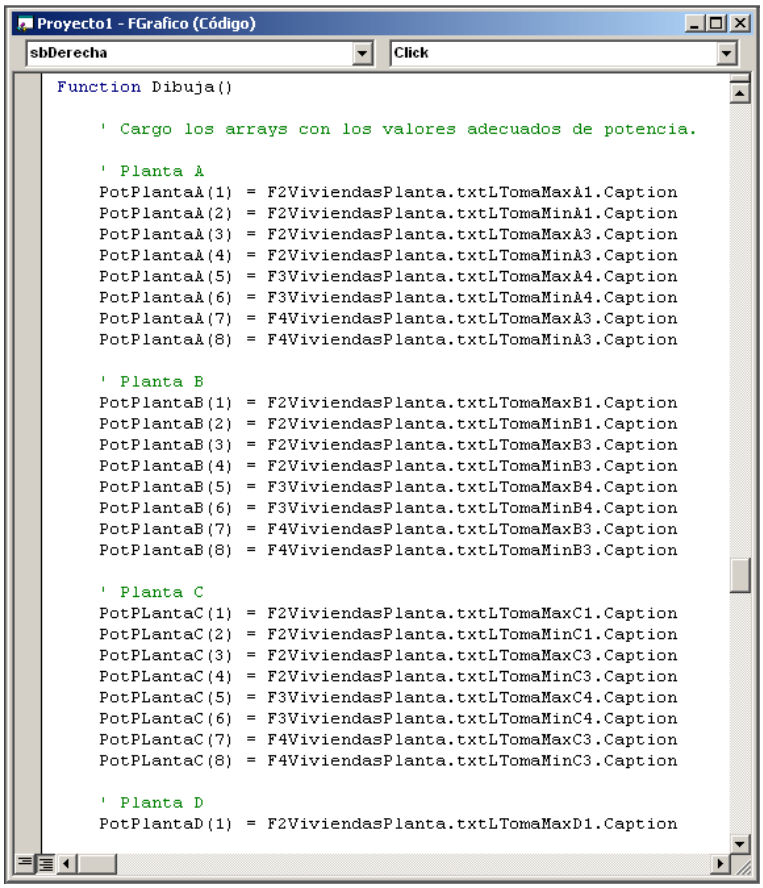

Figura 3.70. *Función Dibuja. Secciones de código para el dibujo*

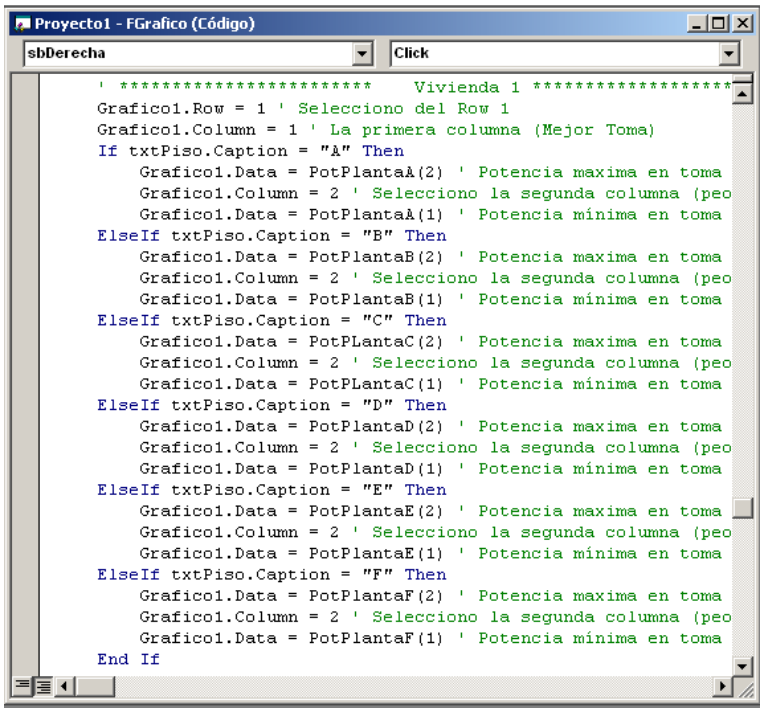

Figura 3.71.*Función Dibuja. Secciones de código para el dibujo*

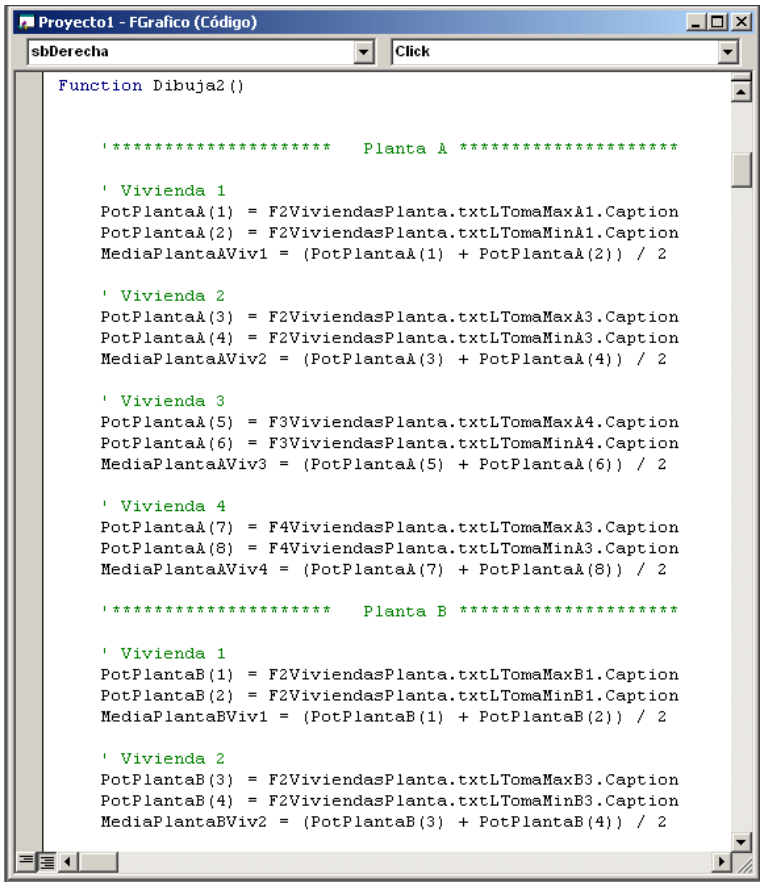

Figura 3.72. *Función Dibuja2. Secciones de código para el dibujo*

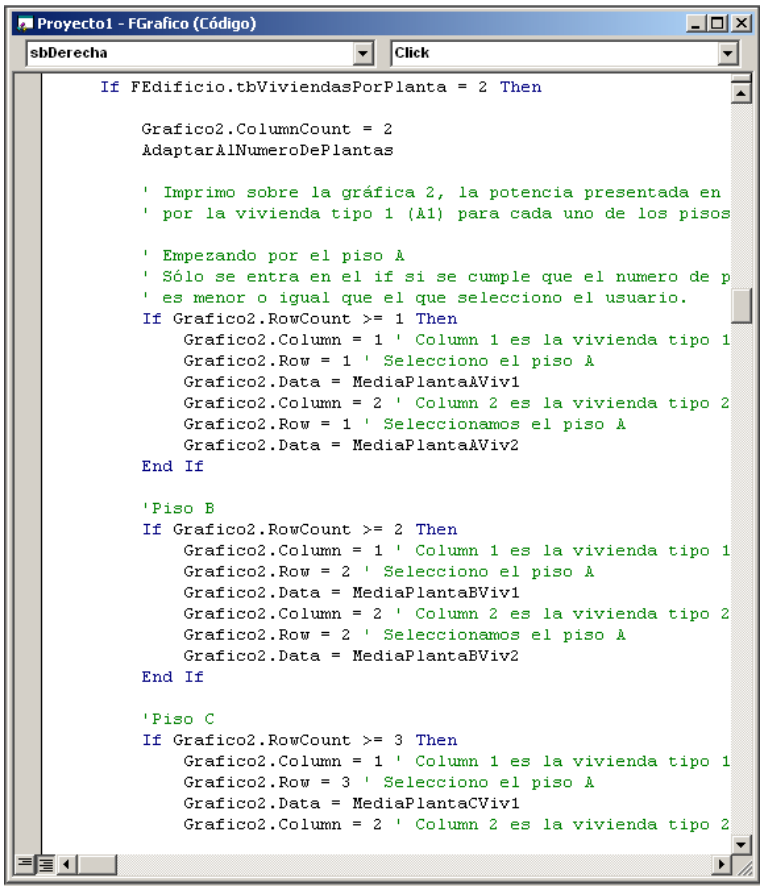

Figura 3.73. *Función Dibuja2. Secciones de código para el dibujo*

A continuación se presentan los formularios encargados en la generación del informe automático. Dichos formularios carecen de complejidad y basan su funcionamiento en completar cadenas de texto con datos particularizados tomados del propio programa. La tarea es sencilla pero tediosa y se prevé que en las futuras versiones de *Calcicat*, se generará la totalidad del informe pedido por el ministerio o bien se generará a partir de macros implementadas en Microsoft Word o procesadores de texto similares. La figura 3.74 remarca un párrafo del informe donde se puede apreciar que lo que se hace es completar cadenas con campos de formularios del programa.

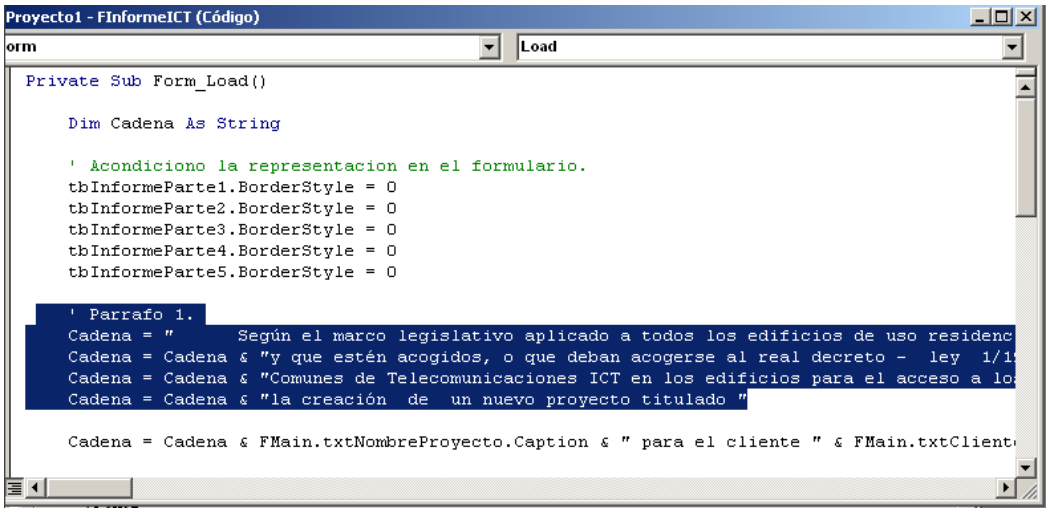

Figura 3.74. *Generación del Informe automático*

Hablaremos ahora de *"Fmain.frm",* el formulario principal que representa el interfaz con el usuario. A dicho formulario se le ha otorgado la misión de generar la barra de herramientas, el asistente de creación cuando así lo solicite el usuario, o el modo experto, en caso contrario, y posibilitar la vista de información del proyecto actual en todo momento (niveles de señal, tomas seleccionadas, monocanales, dispositivos de planta, etc.). Es por ello el pilar básico sobre el que *Calcicat* se sustenta y consecuentemente su correcta programación resultó crucial en la creación de la primera versión del programa. Sus partes fundamentales, groso modo, se resumen en la siguiente tabla.

Función	Finalidad
ConfRedDistribucion()	Configura la red de distribución: Derivadores, distribuidores, tomas y paus.
InicializarVariables()	Declara y asigna variables
PerfilEdificio()	Inicializa las posibilidades de generación del edificio. Es el predecesor de la función CalcularICT del módulo de Datos.
ConfCableado()	Configuración del cableado utilizado
ConfEdificio()	Idem al anterior con el edificio
InformeICT()	Redacta el informe automático particularizando con todos los datos de los que dispone en memoria.
RedDeDistribucion()	Abre la ventana de configuración de la red de distribución.
NivelesDeAzotea()	Abre la ventana de configuración de los niveles de azotea.
NuevoProyecto()	Abre la ventana de configuración del nuevo proyecto.
Guardar()	Se encarga de almacenar todos los datos pertinentes al proyecto actual en el formato deseado por el usuario, nombre y extensión.
CambiarUsuario()	Permite la selección de distintos usuarios que han sido dados de alta en

	una base de datos, el resto no pueden usar el software.
InterfazGrafico()	Utilizado por el programa internamente para completar el aspecto del interfaz gráfico presentado al usuario.
Abrir()	Carga la ventana típica de una aplicación de Windows donde podemos seleccionar proyectos de ICT almacenados y cargarlos
Imprimir()	Imprime la información del proyecto actual.
MnuXXX()	Genera el sistema de menús mostrado
AyudaTotal()	Abre la posibilidad de la ayuda (equivalente al F1 en una aplicación de Windows)
Manual()	Directamente abre el manual de programa.
Expandir()	Carga información de excedencia de formulario.
Internet()	Carga la web oficial del programa

Existen aún más funciones pero carecen de relevancia y por ello se obvian aquí. Si se desea acceder a la totalidad del código se debe cargar el programa desde VB y ejecutar en el modo diseño.

El formulario de opciones "*FOpciones.frm*" presenta una especial relevancia al tratarse de operaciones muy importantes sobre el esquemático. Si bien gran parte de sus funciones son externas, las internas más importantes son:

Función	Objetivo
Atenuaciones()	Genera el esquema de atenuaciones para todas las frecuencias a las que el sistema trabaja.
RepartidorAzotea() (Figura 3.75)	Introduce el efecto de un repartidor en la troncal de bajada.
Graficos()	Genera las gráficas de reparto de señal comentados en puntos anteriores.
Recalcular() (Figura 3.76)	Es una de las funciones más importantes del panel de control ya que, gracias a ella, surten efecto todos los cambios aplicados sobre el esquemático.
Refresco()	Si en algún momento el sistema se saturase, con esta opción se consigue reanudar los cálculos.
SistemaCabecera() (Figura 3.79)	Averigua los monocanales activos del sistema para realizar los distintos cálculos por puntos de frecuencia.
TodasLasFrecuencias()	Calcula el esquemático para todas las frecuencias del sistema.
VerSimbolos()	Expande un formulario que muestra el significado de cada icono en el esquemático.

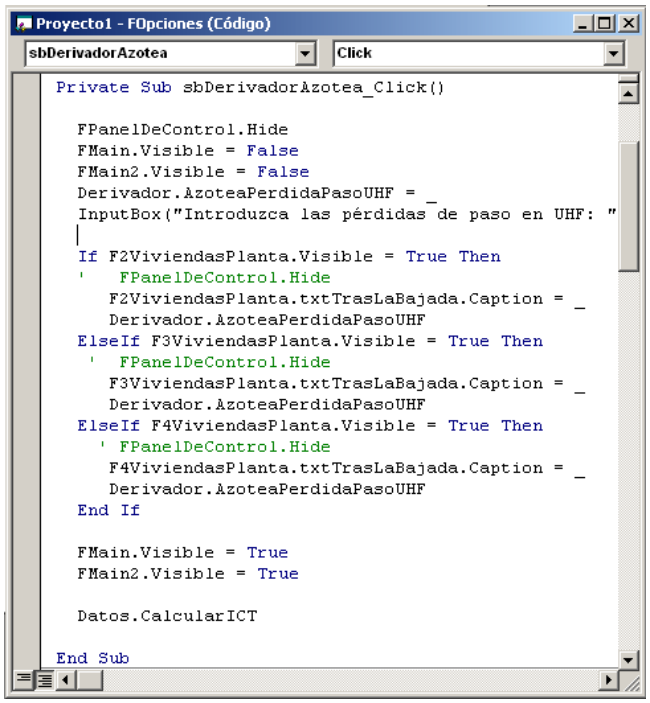

Figura 3.75. *Sección de código de FOpciones (Parte 1)*

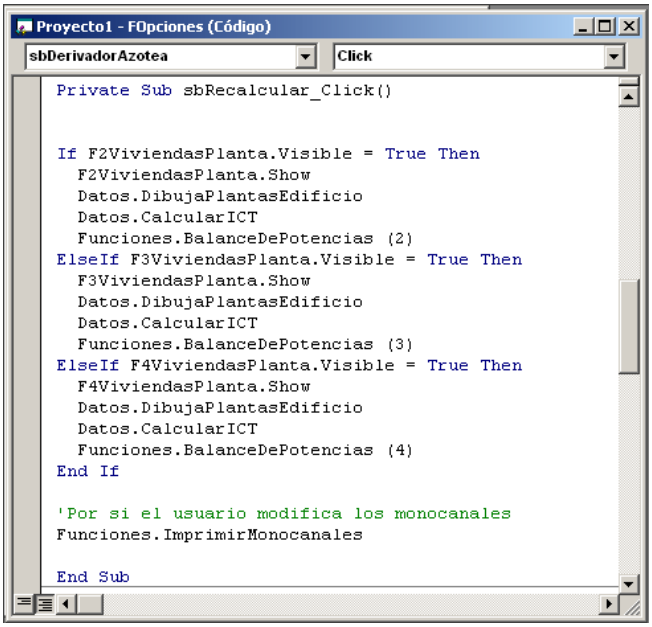

Figura 3.76. *Sección de código de FOpciones (Parte 2)*

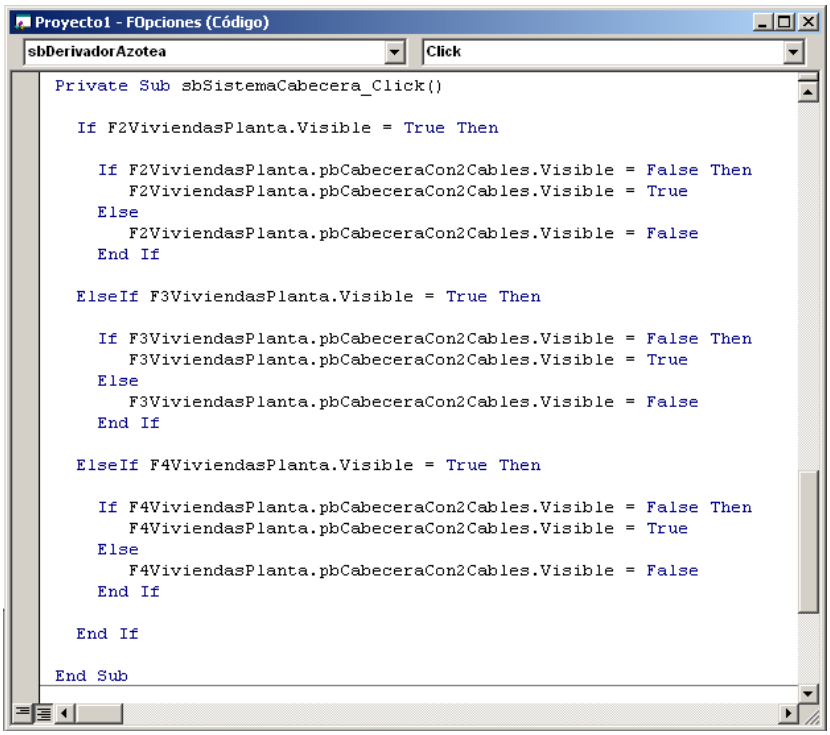

Figura 3.77. *Sección de código de FOpciones (Parte 3)*

Los formularios "*FPanelDeControl.frm*", "*RedDeDistribucion.frm*" y "*FSimbolos.frm*" son importantes, si bien su código no aporta nada nuevo respecto lo explicado anteriormente y por ello se obviará su explicación, pasando a estudiar el formulario encargado de realizar los cálculos para todas las frecuencias "*FTodasLasFrecuencias.frm*".

En él, de nuevo, la parte de cálculo la realiza el módulo datos (figura 3.78), si bien éste formulario ejecuta dicha operación tantas veces como monocanales definidos hayan en el sistema. Por ello cada vez debe introducir un valor de atenuación de cable distinto a través de la variable de acceso pública *Cable.AtCable*. Además, actualmente el usuario es el encargado de ir especificando una por una las diferentes frecuencias a las que el sistema trabaja e indicar para cada una de ellas su correspondiente atenuación en términos de dB/100m. Se está pensando que las próximas versiones incorporen una base de datos de cables y unas funciones que permitan aproximar dichos valores; de este modo dicha operación que a priori resulta bastante tediosa se convertirá en algo sencillo.

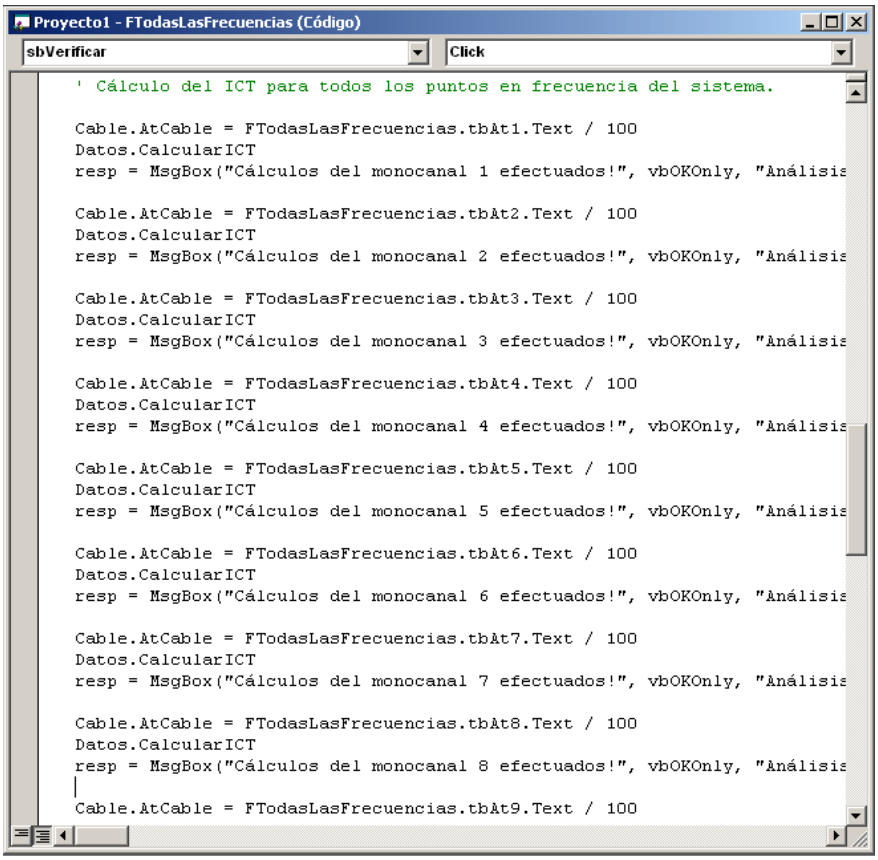

Figura 3.78. *Sección de código de FTodasLasFrecuencias.*

Para finalizar el capítulo, resta estudiar el último de los formularios, "*FTomas.frm*", aunque éste no aporta nada nuevo respecto a los formularios anteriores. A continuación se enumeran sus funciones principales, mayoritariamente conocidas ya por el lector, con el fin de tener una idea general de su funcionamiento interno:

Funciones	Objetivo
Form_Load()	Carga el interfaz gráfico de usuario
PbManual_Clic()	Permite especificar al usuario los datos de longitud de toma corta y larga de modo homogéneo o totalmente particularizado.
Anterior(), Siguiente(), Primero(), Ultimo(), Borrar(), Nuevo(), etc.	Permiten moverse sobre las bases de datos del sistema, añadir nuevos dispositivos, eliminar viejos, etc.
VerificarDatos()	Utilizada por el formulario para almacenar los datos de las tomas en memoria (pérdidas de paso, longitud máxima y mínima, etc)